DOCTORAL THESIS

Green Parallel Metaheuristics:

Design, Implementation, and Evaluation

PhD Thesis Dissertation in Computer Science

Author

Amr Abdelhafez

Directors

Enrique Alba Torres and Gabriel Luque Polo

E. T. S. I. Informática

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA (España)

April 2020

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

AUTOR: Amr Abdelhafez Abdelsamee Galal

http://orcid.org/0000-0003-0469-3791

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

BY NC ND

Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:
http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode
Cualquier parte de esta obra se puede reproducir sin autorización
pero con el reconocimiento y atribución de los autores.
No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es

UNIVERSIDAD
DE MÁLAGA

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS DOCTORAL

Considerando que la presentación de un trabajo hecho por otra persona o la copia de textos, fotos y gráficas sin citar su procedencia se considera plagio, el abajo firmante Don **Amr Abdelhafez** con NIE: X que presenta la tesis doctoral con el título: **Metaheurísticas Paralelas Verdes: Diseño, Implementación y Evaluación**, declara la autoría y asume la originalidad de este trabajo, donde se han utilizado distintas fuentes que han sido todas citadas debidamente en la memoria.

Y para que así conste firmo el presente documento en Málaga a………………..

El autor: ……………………..

UNIVERSIDAD
DE MÁLAGA

Departamento de Lenguajes Y Ciencias de La Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

El Dr. **Enrique Alba Torres** y el Dr. **Gabriel Luque Polo**, pertenecientes al Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga,

Certifican

que, D. **Amr Abdelhafez**, Magíster en Informática por la Universidad de Assuit, ha realizado en el departamento de Lenguajes y Ciencias de la Computación en la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su tesis doctoral (por compendio de artículos) titulada:

Metaheurísticas Paralelas Verdes: Diseño, Implementación y Evaluación

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en la legislación vigente, autorizamos la presentación de esta tesis doctoral en la Universidad de Málaga.

En Málaga, abril de 2020

Fdo: **Enrique Alba Torres y Gabriel Luque Polo**

UNIVERSIDAD
DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD
DE MÁLAGA

For my lovely children:

Marwan, Mazen, and Maram

UNIVERSIDAD
DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD
DE MÁLAGA

Preface

The demand for solving complex and high-dimensional problems has steadily motivated researchers to enhance and purpose new search algorithms. Two major approaches are traditionally used to tackle these problems: exact methods and metaheuristics [8]. Exact methods allow exact solutions to be found but are often impractical as they are extremely time-consuming for real-world problems (large dimension, hardly constrained, multimodal, time-varying problems, etc.). On the other hand, metaheuristics provide sub-optimal/optimal solutions in reasonable time-bound [8, 111]. The metaheuristics field contributes with a wide set of powerful and efficient algorithms for offering practical solutions for optimization problems. The variance of the algorithms in this domain from single-solution to population-based algorithms, equipped with various search operators, promotes these techniques to be powerful in solving optimization and search problems [21]. Most of search and optimization algorithms suffer from the high computational cost and time-consumption curse; in this context, parallel execution for the search process emerged as a very popular option for improving metaheuristics algorithms [8]. Their fields of application range from combinatorial optimization, bioinformatics, and energy to economics, software engineering, etc., whenever fast solutions with high quality are needed [38, 45].

Parallelism comes as a natural way not only to reduce the search time but also to improve the quality of the solutions provided [1, 8]. Their parallel execution reduces the search time and improves the quality of the solution by utilizing the different distributed search information from all the computing units. Parallel and distributed computation is currently an area of intense research activity, motivated by a variety of factors. There has always been a need for the solution of very large computational problems. However, it is only recently that technological advances have raised the possibility of using powerful parallel computers to solve problems not addressed in the past [106]. Most of the modern computing units such as laptops, mobiles, and workstations are equipped with multi-core processing units. These resources present an attractive architecture for designing and implementing efficient distributed algorithms. Using these parallel architectures have many benefits in designing and implementing parallel metaheuristics from many different aspects. In a first aspect, the resulting distributed algorithm has a different design from the sequential algorithm, which numerically springs out providing different search behavior. The second aspect deals with running many processing units in solving one problem, which will lead to overcome the drawbacks of the expensive computational time of metaheuristics and solve the problem faster (speed-up) with still a high-quality numerical solution [9]. The third aspect relates to the merge between parallel metaheuristics and powerful processing units that will allow us to solve large-scale problems and real-world problems that involve big-data faster and more accurately.

Recently, energy efficiency has gained the interest of researchers to optimize computing resources, especially in data centers and clouds [2, 42, 58, 71]. These data centers consume enormous amounts of electrical power. By optimizing these resources, besides saving high operating costs consumed by computing resources, we will save the environment from significant emissions of pollution. Even if electricity is an environmentally friendly form of energy, its generation emits tons of Carbon dioxide (CO_2) and Sulfur dioxide (SO_2) to the environment what increases global

UNIVERSIDAD DE MÁLAGA

warming. These pollutants represent a great threat to humans, animals and plant health [71]. Parallelization (e.g., distribution) is a promising approach for overcoming the overwhelming energy and time consumption values of these methods. Apart from recent approaches in running metaheuristics in parallel, the community still lacks for novel studies comparing and benchmarking the canonical optimization techniques while being running in parallel from an energy point of view. The importance of studying the behavior of parallel algorithms lies in later giving the researchers the ability to develop efficient algorithms that will be energy and time-efficient. Building parallel green computing applications requires the knowledge of the efficiency of the design functionality and the actual energy consumption of each component of the algorithms.

Metaheuristics are being used widely to solve a large range of problems in scientific and real-world problems. However, there are almost no research efforts in studying their internal energy consumption behavior. Most of the search techniques were designed for finding a suitable solution at a reduced computation cost. According to this objective, the numerical performance of the algorithms has been steadily growing. However, the study of the energy consumption of the algorithms has been mostly ignored. Measuring the energy consumption of software still faces many practical and theoretical problems, e.g., the lack of specialized software able to accurately measure energy consumption (hardware independent) [58]. Even the actual building of software that uses this information is quite imprecise [116]. Among the most consuming type of programs we can find out, search techniques when solving complex problems have a prominent place. Search techniques require large computation times and are frequently run to optimize daily activities in cities and factories [42]. All the mentioned above performance and design issues of metaheuristics have motivated our work in this thesis.

Green parallel metaheuristics (GPM) is a new concept we want to introduce in this thesis. It is an idea inspired from two facts: (i) parallel metaheuristics could help as unique tools to solve optimization problems in energy savings applications and sustainability, and (ii) these algorithms themselves run on multiprocessors, clusters, and grids of computers and then consume energy, so they need an energy analysis study for their different implementations over multiprocessors. The context for this thesis is to make a modern and competitive effort to extend the capability of present intelligent search optimization techniques. Analyzing the different sequential and parallel metaheuristics considering its energy consumption requires a deep investigation of the numerical performance, the execution time for efficient future designing to these algorithms. We present a study of the speed-up of the different parallel implementations over a different number of computing units. Moreover, we analyze and compare the energy consumption and numerical performance of the sequential/parallel algorithms and their components: a jump in the efficiency of the algorithms that would probably have a wide impact on the domains involved.

UNIVERSIDAD
DE MÁLAGA

Acknowledgements

This thesis is the outcome of my Ph.D. studies in Spain for about four years, although I have not walked all that way alone.

Firstly, I would like to express my gratitude to my finding agency, the **Egyptian government**, represented in the Egyptian Institute (El Instituto Egipcio en Madrid) in Spain. My sincerest thanks go to my supervisors Prof. Dr. **Enrique Alba** and Dr. **Gabriel Luque** for their invaluable advice and comments for the thesis during my study. Prof. Alba always provided me with ideas and hints, which usually emerged from long scientific discussions. Dr. Luque always was the eye that revised and suggested solutions for the technical issues that we have encountered during the study.

I would also thank the **NEO** group members, whom I am glad that I met and shared time with them during my stay in the laboratory. In particular, I would mention: Andrés Camero, Christian Cintrano, Javier Ferrer, José Á. Morell, Rubén Saborido. Supported with their help, I want to thank **my family** for all that they provided to me during my stay in Spain. **My children**, I am always encouraged and powered by your love. Unlimited thanks go to my wife **Mona Elbadry** for her endless help, patience, encouragement, and support with soul.

Amr Abdelhafez

UNIVERSIDAD DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD DE MÁLAGA

Contents

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

List of Figures

UNIVERSIDAD DE MÁLAGA

List of Tables

UNIVERSIDAD
DE MÁLAGA

Chapter 1

Introduction

In this chapter, we overview the domain of metaheuristics and present its main classes and components. We present the parallel models and communication schemes used for running metaheuristics in parallel. Further, we outline the energy efficiency concepts. Finally, we present the motivation, and define the objectives and goals of this work, and outline the thesis organization.

1.1 Metaheuristics

Optimization problems occur in the real-world industry, science, engineering, economics, and other real-life activities [47, 69]. These problems are usually complex and difficult to solve by exact search methods [9]. The deep need for solving these problems has activated the emergence of a new class of algorithms called "Metaheuristics" [45]. Metaheuristics proved to solve large problems faster in a reasonable time-bound [111]. The past five decades have witnessed the emergence of many metaheuristics algorithms, many of these algorithms imitate a phenomenon in nature or industry, e.g., genetic evolution in nature and annealing process in metallurgy. Metaheuristics algorithms vary in their design and search behavior, thus, showing different performance in solving problems in different research fields. Metaheuristics are mainly composed of two main categories of algorithms, *Trajectory*, and *Population-based* algorithms [8].

Trajectory methods follow a track of improvement to a single solution to explore the search space. This iterative improvement is usually combined with other search operators for an efficient exploration or escaping from local minima. Population-based algorithms go across the search space with a population of solutions. Each solution represents a tentative optimal solution. The population of solutions is usually initialized randomly. At every iteration, a new population is generated using the search operators of the method. After the evaluation phase (using selection procedures), some solutions are selected to proceed for the new phase. This procedure is repeated until the termination condition met (e.g., reaching a maximum number of evaluations or a solution with a specific quality).

In the following section, we explain in detail three important and widely-used metaheuristics: Simulated Annealing, Variable Neighborhood Search, and Genetic Algorithm. These methods are very common search algorithms used by researchers and industry [18, 23, 46, 69].

UNIVERSIDAD DE MÁLAGA

1.1.1 Simulated Annealing

One of the most famous and widely-used trajectory methods is Simulated Annealing (SA). Guided with the inspiration of the annealing process in metallurgy, Kirkpatrick et al. [68] proposed the SA method for solving unconstrained and global optimization problems. SA models the physical process of heating and annealing the metals by decreasing the temperature with a predefined schedule, thus improving the quality of the metal and minimizing the system energy loss. SA manages the search process by considering a single solution, regularly modified with an appropriate cooling schedule (annealing) for generating a better one in terms of its quality. Algorithm 1 presents the canonical SA procedure, *r* and *L* refers to the cooling ratio and the integer temperature length, respectively [125].

Algorithm 1 Simulated Annealing Algorithm [125]

Initialization. Solution S, Temperature T > 0
while $not\ stop - condition$ **do**
    for `i = 1 to L` **do**
        Pick a random neighbor S of S;
        $\Delta \leftarrow (cost(S') - cost(S))$;
        if $\Delta \leq 0$ **then**
            $S \leftarrow S'$
        else
            if $\Delta \geq 0$ **then**
                $S \leftarrow S'$ with probability $e^{\Delta/T}$;
            end if
        end if
    end for
    $T \leftarrow rT$ (reduce temperature);
end while

Due to SA design functionality by allowing occasional uphill moves, SA is able to avoid being trapped in local minima [111].

1.1.2 Variable Neighborhood Search

Variable Neighborhood Search (VNS) [84] is a widely applicable metaheuristic algorithm used for solving global and combinatorial optimization problems based on the systematic change of neighborhood to generate trials from the current solution. The basic VNS explores the search space of the problem by generating trial solutions in the neighborhood of the current solution. VNS yields a single-solution and jumps from the current solution to another if an improvement is found. This search procedure makes of VNS a simple and effective metaheuristic for solving search problems [54, 97]. VNS has been employed successfully in solving numerous problems in real-world problems and industry, e.g., Global optimization, Traveling Ready-Mixed Concrete Delivery Problems, Vehicle Routing Problems [51, 91, 122].

UNIVERSIDAD
DE MÁLAGA

VNS considers a set of neighborhood structures to be used in a systematic way to conduct a search through the solution space. The design of the VNS promotes it able to escape from local minima and from the valleys which contain them. These set of neighborhood structures used in the VNS acts as the main difference between VNS and basic local search (a single neighborhood in local search) [83]. The neighborhood $N_k(x)$ indicates the position of the tentative solutions in the k^{th} neighborhood of x, as shown in equation 1.1.

$$N_k(x) = \{y \in S \mid \rho_k(x, y) \leq r_k\} \tag{1.1}$$

where r_k is the radius (size) of the neighborhood k and ρ_k is a metric. The geometry of neighborhood structures is playing a leading role in the efficiency of VNS [83]. Algorithm 2 presents the canonical VNS procedure; the shaking procedure refers to generating solutions randomly from the neighborhood of the current solution x.

Algorithm 2 Variable Neighborhood Search [54]

```
Initialization. Select the set of neighborhood structures N_k, Initial solution x;
while not stop − condition do
    repeat
        Shaking: x' ← Shake(x, k)
        Local search: x'' ← Improve(x')
        Change neighbourhood: x ← NeighbourhoodChange(x, x'', k)
    until k = k_max
end while
```

1.1.3 Genetic Algorithm

Inspired by evolution and natural genetic mating, Genetic Algorithms (GAs) [47] are well-known search algorithms used widely for solving optimization and real-world problems. GAs explore the search space of the target problem using a population of solutions. These solutions are mating and evolving during the search using the principles of evolution and natural genetics. GAs proved to be capable of searching and providing optimal or sub-optimal solutions in complex and multimodal search spaces [8, 43, 111].

Genetic Algorithm (GA) starts with a randomly generated set of individuals, called a *population*. Each *individual* (chromosome plus fitness) represents a possible tentative solution. Each *chromosome* is composed of an array of genes depending on the dimension of the problem solved. The *fitness* (optimized objective) function is used to evaluate the quality of every individual in relation to the rest. Genetic *operators* (usually selection, crossover, mutation, and replacement) are used to generate new solutions for the next generation. This process is performed until the stopping criterion is met (maximum number of fitness evaluations or a find a solution of good quality). Algorithm 3 provides the pseudocode of the so called panmictic GA, where one population is considered and being run in one computer.

The term *panmictic* means that all the individuals in the same single population can probably mate to the rest, i.e., there is no restriction to their interactions. In *structured* GAs [10], the individuals are geographically separated, and interactions occur only inside these isolated neighborhoods.

UNIVERSIDAD DE MÁLAGA

Algorithm 3 The Canonical Genetic Algorithm

Initialization. Generate an initial population P randomly.
Evaluation. Evaluate the individuals in P.
while $not\ stop - condition$ **do**
    P' := Crossover operator [P];
    P'' := Mutation operator [P'];
    Evaluate fitness [P''];
    P''' := Select [P''];
end while

Multi-population GAs such as distributed GAs (dGAs) is a typical example of structured GAs [8]. A distributed GA (dGA) can indeed be run in parallel on different cores (or not), but this refers to its physical execution, not to the design of the algorithm (as *distributed* points out).
The canonical GA code is composed of the following components:

- **Fitness evaluation:** This operator is one of the main GA components since it is used for calculating the quality of solutions and guiding the search. A fitness value indicates how close is a solution to the optimal solution. The computational resources required by this operation highly depend on the problem being solved. Other problem characteristics, e.g., multi-objective optimization and dynamic problems have a strong relation to fitness computation, not to mention parallelism, that comes handy in many cases because of its high computational demands [2]. It is now clear the importance of the study for energy consumption of evaluations apart.
- **Genetic operators:** Usually the variation operators of a GA are used to generate new solutions based on the existing ones. The active variation operators in a GA are crossover and mutation. The aim of these operations is to modify the current population to get a new one. The crossover (or recombination) operator combines two or more different solutions to generate new solutions, while the mutation operator modifies the solution by changing its genes to generate a new one. The main role of the mutation operator is to add genetic diversity inside the population, thus preventing the algorithm from converging to a local optimum. Genetic operators are substantial to GAs; they deserve an energy profiling study.
- **Housekeeping:** This term (in our study) refers to all the rest of the tasks of the algorithm, e.g., initialization, selection, and I/O operations. These operations have fewer computations and instructions to be executed compared to the previous operators.

1.2 Parallel Metaheuristics Concepts

Metaheuristics (like most search and optimization methods when solving real problems) suffer from high computational cost and are time-consuming. In a fixed time-bound, due to its stochastic nature, metaheuristics do not guarantee to arrive at an optimal solution for most optimization problems [61]. Many approaches and advances were proposed to overcome these drawbacks. One popular approach is running the algorithms in parallel (e.g., distributed) [8]). The new resulting distributed algorithms reduce the search time and exhibit new search characteristics which differ from the sequential ones [3, 5, 15].

UNIVERSIDAD DE MÁLAGA

The parallel execution (physically independent set of processing units) will reduce the execution time and even could allow an improvement in the quality of the solutions, by sharing the knowledge during the search between different sub-populations [1, 14]. Most of the modern computing devices such as laptops, mobiles, and workstations are equipped with multi-core processing units. Using these parallel architectures have many benefits in implementing and designing parallel metaheuristics from many different aspects. As a first aspect, the parallel technique usually has a different design from the sequential algorithm, what numerically springs out providing different search behaviors. As a second aspect, running over many processing units for solving one problem could lead to overcome the drawbacks of the expensive computational time of GAs and solve the problem faster with still high-quality numerical solutions. As a third aspect, the combination of parallel metaheuristics and powerful processing units will allow us to solve large-scale and real-world problems that involve big-data, faster and more accurately [111].

In literature, there are many varied parallel strategies for running the two categories of metaheuristics (population-based and trajectory-based) algorithms in parallel. According to the source of parallelism, there are three parallel metaheuristics strategies [33] :

- **Low-level parallelism**, this approach considers running the computations in parallel aiming at speeding the search process. This type of parallelism makes an attempt to reduce the execution time, but still, exhibit the same search performance of the sequential algorithm (no improvement in the exploration or the exploitation behavior). A typical model for this parallel approach is the basic master-slave model [26].

- **Data and decision variables decomposition**. In this parallel approach, a search space (domain) decomposition is considered by dividing the search space into smaller regions, thus applying the sequential metaheuristic on each region and the variables outside the subset are considered fixed. Domain decomposition has been employed successfully in designing and running metaheuristics in parallel [115].

- **Multiple concurrent explorations**. Parallelism is obtained from multiple-walks into the solution space concurrently. Each concurrent thread may or may not execute the same heuristic method. These distributed threads communicate during the search for knowledge sharing. An example of this parallel approach is the distributed run of the GA [1].

In the past three decades, many kinds of research were performed using these different parallel approaches. However, still, there are limited efforts in investigating the efficiency and efficacy of these approaches. The importance of studying these different models lies in building future efficient green search algorithms.

1.2.1 Communication Schemes

Metaheuristics could be parallelized by dividing the tasks of the algorithm distributed over different processors. These different processors communicate to share search knowledge and data. A common type of Parallel GAs (PGAs) is the *island model*, also known as the *distributed model* [8]. In this model, the global population is divided into sub-populations (islands) distributed over different processors. These islands run a basic GA each, and after a predefined interval they communicate in order to exchange the search knowledge; e.g. individuals or parameters. The communi-

UNIVERSIDAD DE MÁLAGA

cation between the islands is called “Migration”. The migration operator is exchanging individuals between the sub-populations of the dGA. This process aims at improving the gene pool and hence increases the diversity and accelerate the convergence of the algorithm. Previous researches [13] reported on the typical behavior of dGAs and determined that it is strongly influenced by this migration mechanism. The performance of dGA is then usually affected by the information exchanged and the time interval for this exchange [30]. The important parameters of migration are the migration interval, migration rate, the method for selecting migrating individuals, the method for replacing an individual with the new one in the receiving sub-population, and the topology type [66].

Our migration topology considered in the thesis is based on the uni-directional ring topology analyzed in many works, e.g., [17, 103, 107], where it was ranked as a good topology out of the fourteen different migration topologies studied there. In this topology, the communications between islands are done on a unidirectional ring, and thus an island can send and receive migrants only from its next and previous neighbors, respectively. The ring topology ensures local communications between sub-populations and smooth propagation of good information at a pace that it does not make the algorithm converge too fast, as depicted in Figure 1.1, and keeps low the overall communication effort of the dGA.

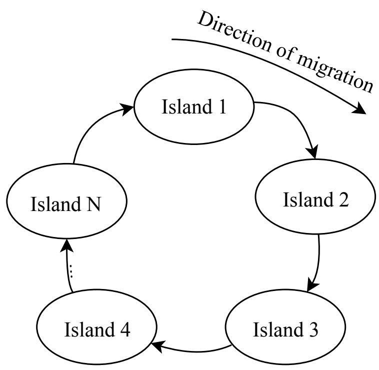


Figure 1.1: Unidirectional ring topology for migration among sub-populations

In our studies, each sub-population sends and receives the best individual from the neighbor sub-population, and replaces the worst one in his pool with the most recent incoming one. The benefit of this design is that migration occurs locally between adjacent populations in the ring. This yields local smooth exploration and exploitation of the search space, while globally the separate sub-populations are free to explore different types of regions independently [79].

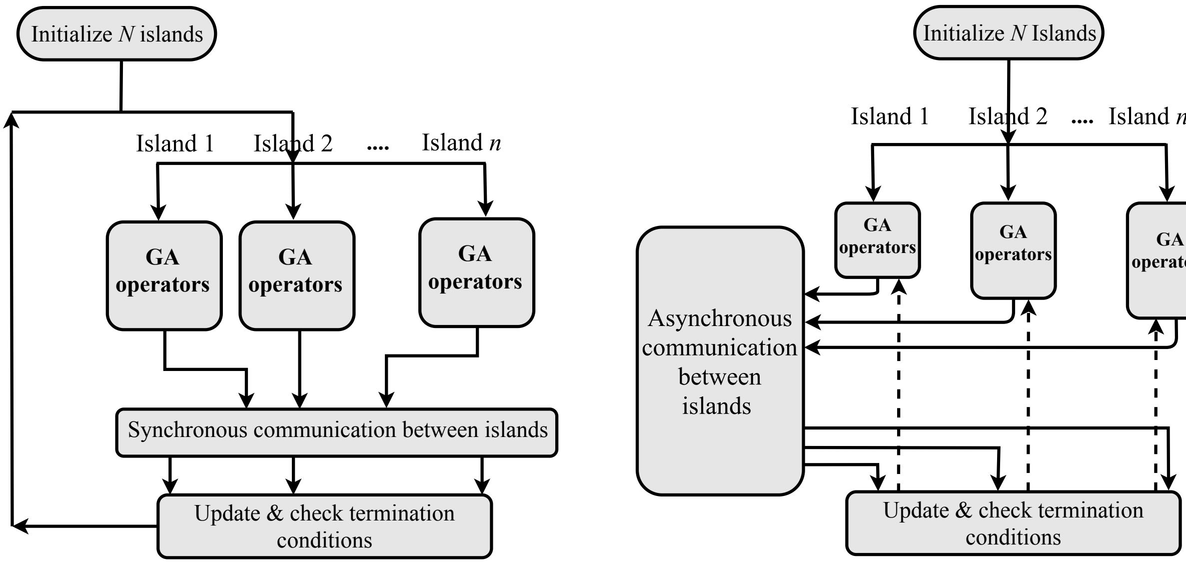
We express the migration interval in terms of the number of evaluations made in a sub-population. The migration interval was chosen to let every process make sufficient exploration on its own between communications. In addition, the migration rate was determined as a fixed number of the best individuals in the population in every interval, to allow a more structured study.

1.2.2 Synchronous and Asynchronous Implementations

The parallel execution with potentially different communication schemes promotes metaheuristics with new search characteristics. In literature, Synchronous and Asynchronous are the most

UNIVERSIDAD DE MÁLAGA

commonly used communication schemes in parallel algorithms. The sync approach has a synchronization point every migration interval, where communications happen [6]. At this point, all the processes should wait and block till all processes reach this same point of their code. So, it is highly probable that (at every communication point) there are some idle processes waiting for the rest to arrive and communicate results. On the contrary, in the async approach, there are no such synchronization points, since the sub-populations include the received individuals whenever it is possible, then avoiding any waiting [14]. So there are no idle processes by construction. The design of the async dGA is developed to enable efficient interactions and promote the overlapping of computations and communications [74]. In our studies, Sync and Async implementations addressed run governed by the same algorithm parameters, with the only difference being in the communication scheme. Figure 1.2 shows the differences between the two schemes.



(a) Synchronous scheme (b) Asynchronous scheme

Figure 1.2: Functional diagram for the sync/async dGA schemes

1.3 Energy Efficient Algorithms

The variance of different algorithms equipped with multiple search operators gives metaheuristics their wide applicability for solving optimization and search problems [4, 111]. The basic versions of these algorithms were designed for running in sequential with the main objective of finding a suitable solution and reducing the computation cost. According to this objective, the numerical performance of the algorithms has been steadily growing. However, the efficiency and compatibility of the parallel execution with the energy consumption of the algorithm has been left unattended.

Nowadays, the higher electricity consumption and carbon dioxide emission are motivating scientists and society for looking for greener applications (that consume less energy). The goal is to

UNIVERSIDAD DE MÁLAGA

progress towards a new design of algorithms that adds energy efficiency in synergy with numerical performance. Two main outcomes can be obtained by improving the energy efficiency of the search algorithms. The first one is that energy-efficient algorithms will reduce the environmental impact. The second one relates to a faster search process that also improves reliability (less heat-overhead is produced by machines, thus less cooling system costs and noise) [87].

Our approach in this thesis is to gather energy consumption values of the different metaheuristics at run-time while using a software approach for collecting the energy consumption values. We consider a system monitoring library (at the programming-language level) to make use of CPU counters, called RAPL [35]. Most of the previous studies that consider measuring energy consumption relied on wall-mounted meters or on-board measurement sensors [67, 95]. Wall-mounted meters measure the energy consumption of the whole system, not the running process. On-board measurement sensors vary on type and method of estimating energy consumption. The energy consumption values obtained by these methods are usually combined with the high overhead of the other running processes and the thermal heating of the machine [20, 52, 92]. RAPL estimates the energy consumption of a running process, in real-time, based on information collected from the Model-specific register (MSR) counters (e.g., CPU, RAM) through the operating system. Recent studies that investigate RAPL efficiency confirmed that RAPL counters are highly accurate [101] and highly correlated with plug power with negligible performance overhead [64].

1.4 Multiprocessors Architecture

Most of modern computing units such as laptops, mobiles, and workstations are equipped with multi-core processing units. These resources present an attractive architecture for designing and implementing efficient distributed algorithms. These modern architectures differ from the single-processor systems, since it may incorporate such as processor-array, super-scalar, or multi-threading [28, 44]. Therefore, the parallel performance of the algorithms running in multiprocessors is highly correlated to the number of cores and the cache size of the parallel platform; that consequence is explained by Amdahl's law [57]. Thus the performance achieved by using a multi-core system strongly depends on the design of the parallel algorithms and their implementation (software used) [57, 100].

For our studies, we consider a multicore machine with 32 cores. This choice of a modern and commonly-used system shall enrich the existing literature for multicore systems against the enormous existing cluster system studies. In a homogeneous multicore machine, all the computing nodes have the same frequency, which means there are no lagging for waiting for slower nodes. Also, using a shared-memory system will reduce communication costs. Based on these factors, the parallel execution of the algorithms under these systems will show different performance profiles. In the up-to-date literature, there are few pieces of research studying the efficiency of the algorithms in multiprocessors [1]. Our analyses will help in developing a novel body of knowledge on algorithms running in shared-memory multiprocessors (versus the overwhelming literature oriented to distributed memory clusters), something useful for researchers, beginners, and final users of these techniques.

UNIVERSIDAD
DE MÁLAGA

1.5 Thesis Methodology

In this section, we explain the motivation, define objectives and goals of this work, and present the thesis organization.

1.5.1 Motivation

Optimization arises everywhere in the industrial and engineering fields with complex and time-consuming problems to be solved [89]. Exact search techniques cannot afford practical solutions for most of real-life problems in reasonable time-bound [8]. Metaheuristics proved to be efficient solvers for such problems in terms of solution quality, however, they require unreasonable time and energy consumption values. Parallelization approaches exist as the solution for reducing the overwhelming energy and time consumption values of search techniques. In the past three decades, many studies were done to investigate and validate the performance of the parallel execution of metaheuristics. However, this field requires more studies that compare and benchmark the search techniques while being running in parallel.

Metaheuristics encompass a wide range of different algorithms, varying from single-solution to population-based algorithms [8]. The variance of different algorithms equipped with multiple search operators gives metaheuristics their wide applicability for solving optimization and search problems [111]. The basic search technique of these algorithms was designed for running in sequential to find a suitable solution and reduce the computation cost. According to this objective, the numerical performance of the algorithms has been steadily grown. However, the efficiency and compatibility of the parallel execution with the energy consumption of the algorithm has been left unattended [2].

The parallel run of metaheuristics looks promising from the design-point of view. Previous researches [8, 15] confirmed the different search characteristics and reduced run-time of parallel metaheuristics. The compatibility of the parallel run and the design-functionality of the algorithm require deep run-time analysis. Most of metaheuristics operators were designed to run sequentially on a stochastic manner, thus running the algorithm in parallel will affect the design-functionality of these search operators. Besides the mentioned-before prospective parallel performance issues, the energy consumption and its relation to the distributed algorithm performance have been left unattended in most of the existing literature. The efficient utilization of the full set of processors in a multiprocessor is also a big challenge today as it has an impact in reducing the energy needed to maintain computing centers using these machines.

The communication between the parallel/distributed algorithms usually involves selecting some individuals from one sub-population and sending them to other islands. The migrated individuals are integrated into the local sub-population according to predefined acceptance criteria. The communication type and the acceptance criteria of the migrated individuals are very important in determining the convergence behavior of the algorithm [12]. Running algorithms on a multiprocessor system can potentially vanish the numerical and run times differences between the different communication schemes. Besides, statistical comparisons between the different communication schemes increase the body of knowledge in this domain, allowing researchers and final users to build new efficient search techniques that fit with their goals.

UNIVERSIDAD DE MÁLAGA

In today's world, energy efficiency is an important topic in all scientific areas. In Computer Science, a new domain called *Green Computing* [60] has emerged to deal with the efficient use of computing resources to reduce the environmental footprint. The electricity consumption of computing devices has a large impact in our current digital era, and it will increase in the next years. Electricity generation and distribution is combined with an enormous amount of pollutants. These pollutants increase global warming and represent a threat to any living being [82].

Nowadays, the higher electricity consumption and carbon dioxide emission are motivating scientists and society for looking for greener applications (that consume less energy) [2, 58, 60]. The goal is to progress towards a new design of algorithms that adds energy efficiency in synergy with numerical performance. This interconnection of energy consumption and the numerical efficiency of the algorithm suggests a high interest in studying and analyzing current state-of-the-art search algorithms by considering their energy consumption. Overall, the literature on analyzing energy consumption of metaheuristics exists, though it is very limited yet. We can find several articles on nearby fields, like applying metaheuristics for problems dealing with energy minimization in a target scenario [20, 67]. However, that approach is not targeting the energy consumption of the algorithm itself but using the algorithm to solve energy-related problems. In the associated literature, there are almost no studies for the energy consumption of sequential and parallel metaheuristics and their components, what would allow the next phase of smart use of this information to build efficient algorithms with lower consumption for a similar numerical result.

1.5.2 Objectives and Goals

Our objectives in this proposal are to present green parallel metaheuristic algorithm concepts, for better future designs and energy-efficient parallel algorithms, over modern computing platforms. We concentrate mainly on two research approaches directly connected to parallel metaheuristics. The first approach is analyzing the behavior of parallel metaheuristic algorithms (e.g., numerical performance, speed up, and execution time) while considering different parallel models and communication schemes over multiprocessors. The second approach is analyzing the energy consumption of the different metaheuristics algorithms (parallel/sequential) for better future deigns of efficient search algorithms.

The concrete sub-goals aiming of this thesis are the following ones:

- **O1.** Analyze the performance of different parallel metaheuristics implementations over modern computing platforms (multiprocessors, in this thesis).
- **O2.** Study the effect of synchronism on PGAs, presenting their different implementations to compare their expected similarities and differences over multiprocessors.
- **O3.** Analyze the energy consumption of sequential and parallel metaheuristics and their components.
- **O4.** Develop a novel body of knowledge on metaheuristics running in shared-memory multiprocessors and test them on academic benchmarks.

For O1, we perform many extensive performance analyses for the algorithms by considering multiple different termination conditions. Such combined experiments analyze the parallel performance of the addressed algorithms and show the combinatorics of parallel run and metaheuristics.

UNIVERSIDAD DE MÁLAGA

We presented the numerical results and speed-up of the different metaheuristics while changing the number of computing units used. This research approach will highlight the differences/similarities of the considered algorithms from the solution-quality and execution times points of view. The outcome of these experiments will enrich the knowledge for future studies comparisons with cluster-based distributed algorithms.

For O2, we investigate the effect of synchronizing communications of PGAs over modern shared-memory multiprocessors. We consider the master-slave model along with synchronous and asynchronous dGAs, presenting their different designs and expected similarities when running in a number of cores ranging from one to 32 cores. We consider analyzing the results of many problems with several dimension sizes. These problems possess different search spaces and different dimension sizes. Thus, we could show the effect of the synchronism on the algorithms, using a different number of problems and dimensions over a various number of cores.

For O3, we study the energy consumption and execution time behavior of both sequential and parallel metaheuristics, in a reasonably wide energy analysis of its components under different computer communication schemes. The results and discussions presented will show the relation between using more computing units on numerical performance, energy-consumption, and execution time behavior. Such results represent an extensive study on the performance of the state-of-the-art metaheuristic algorithms. The potential impact of these results in building new techniques that fit the aims of green computing will also link to a line of research for making algorithms more efficient as a piece of software running on a computer. This point of view is not so present in the associated literature, but a fundamental milestone for formulating high-quality research.

For O4, we present a wide set of experiments to analyze the combinatorics between metaheuristics and solving optimization problems while being run in parallel. The combined outcome of these experiments shows the numerical performance, energy consumption, and execution time of the parallel optimization algorithms. For our studies, we consider a multicore machine with 32 cores. Our research approach offers knowledge that will help to reduce the energy consumption of the algorithms and their components thus saving the environment from being polluted by electricity generation and distribution. The overall outcome of these studies builds a guide for future designs of efficient and energy-aware optimization techniques.

1.5.3 Contributions and Publications

In this section, we present the contributions and scientific works published during the years of this thesis. These publications were the result of our experiments, which we have done to satisfy the objectives of the thesis. These publications were evaluated by experienced experts in the academic field, to ensure their validity and impact in the computer science field. In total, there are five scientific publications: three in international indexed journals and two in international conferences. Detailed as follows:

[1] Amr Abdelhafez and Enrique Alba. **Speed-up of synchronous and asynchronous distributed genetic algorithms: A first common approach on multiprocessors.** 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2677-2682, 2017.

UNIVERSIDAD DE MÁLAGA

[2] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors.** Swarm and Evolutionary Computation, 49:147-157, 2019.

[3] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **A component-based study of energy consumption for sequential and parallel genetic algorithms.** The Journal of Supercomputing, 75(10):6194-6219, 2019.

[4] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Analyzing the energy consumption of sequential and parallel metaheuristics.** 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019.

[5] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Parallel Execution Combinatorics with Metaheuristics: Comparative Study.** Swarm and Evolutionary Computation, 2020.
DOI: https://doi.org/10.1016/j.swevo.2020.100692

The first and second publications were the result of the experiments that we have done to satisfy the first and second objectives of the thesis, respectively. In these publications, we investigated the parallel performance of GAs over multiprocessors considering three different parallel models: Master/slave, synchronous, and asynchronous distributed models. We presented the effect of synchronism on PGAs by showing the similarities and differences of these different communications schemes.

The third and fourth publications were the result of the experiments that we have done to satisfy the first and third objectives of the thesis. We analyzed the energy consumption of GA components one by one, over a varied number of problems and dimensions. We have used various test problems with different dimensions to study the effect of the size of the problem on the energy consumption of different GA components. Further, we extended this analysis to a dGA with two different communication schemes (synchronous and asynchronous). The fourth publication represents an extensive study of the energy consumption features of three different metaheuristics and their distributed counterparts. We considered a varied set of problem features (dimensionality, search landscape, multi-modality, etc.) so that meaningful conclusions are feasible.

The fifth publication was the result of the study that we have done to satisfy the first and fourth objectives of the thesis. This publication represents an extensive analysis of the parallel combinatorics and optimization. We presented a comparative study to three widely-used metaheuristics and their distributed counterparts. Our algorithms include trajectory-based and population-based approaches and their distributed counterparts. We considered running our experiments using the two main termination conditions found in literature: (1) considering a fixed number of function evaluations and (2) arriving at a predefined fitness value. Such combined experiments show the combinatorics of parallel run and metaheuristics while changing the number of computing units used; thus, differences/similarities of the considered algorithms from the solution-quality and energy consumption emerge and discussed. The significance of measuring the energy consumption of the algorithms is considering the aspect of green-computing while designing and running the algorithms.

UNIVERSIDAD DE MÁLAGA

1.5.4 Organization

This thesis is written in the form of a compendium of publications. The organization of this dissertation is as follows. Chapter 2 reviews the state of the art in the parallel and distributed metaheuristics area. Chapter 3 presents the compendium of the published works, where we present the main contributions and summarize the published articles. We conclude our thesis in Chapter 4 with concluding remarks and future works.

UNIVERSIDAD DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD DE MÁLAGA

Chapter 2

State of the Art

Real-life problems keep rising and growing in size and complexity (e.g., optimization problems, scheduling, etc.), combined with the emerging of new search methods for facing these problems. In today's world, using exact search methods is not a reliable way to solve most of the problems. Most of the exact search methods are not capable of providing practical solutions for real-life problems. Evolutionary Computation (EC) has gained much attention from the researchers in the past decades, due to its importance in solving real-life numerical optimization problems. These algorithms are considered as a practical solver for most of the real-life problems [8, 111]. These methods usually consume a considerable amount of time and energy when used in daily tasks workstations. Therefore, the analysis of the performance of these methods is crucial for building real green computing applications.

The main urge for turning to the parallelization of metaheuristics is reducing the computation effort and providing a better speed-ups. Nowadays, there is a fast advancement in designing and proposing new processor architectures (e.g., Multiprocessors, and Mobile Processors). These new computing architectures have encouraged the use of the parallel approach in solving daily real-life problems. Thus appears the demand for analyzing the performance and energy consumption of these algorithms when run under these architectures. This chapter briefly reviews the recent in state of the art in the parallel and distributed metaheuristics and energy efficiency areas.

2.1 Parallel Metaheuristics: Strategies and Advances

As a common approach for optimizing current era problems, metaheuristics proved to be a reliable choice in solving problems appears on scientific and industrial fields. Several studies have been done over the past two decades for benchmark and analyze the parallel behavior of metaheuristics algorithms. In this section, we review some of these efforts, exploring the parallel approaches employed for the metaheuristics algorithms.

Metaheuristics have gained much attention from the researchers in the past decades, due to its importance in solving real-life numerical optimization problems. The design of the parallel model and the cost of synchronizing the communications can highly affect the performance of PGAs. Every migration step in the sync implementation acts as a global barrier where all processes share knowledge. The delay of any process (for any reason, such as network, computing, or I/O opera-

UNIVERSIDAD DE MÁLAGA

tions) will affect all the other processes and the whole search process. Such delay will possibility increase as the number of processes also increases. Therefore, the async implementation looks, in theory, like a promising approach designed to get a higher performance because of its non-blocking communication scheme. In this present section, we review some of the works that investigated the performance of the different dGAs models and study effect of synchronism on the efficiency and accuracy of Evolutionary Algorithms (EAs).

2.1.1 Master-slave Model

Master-slave presents as a common paradigm used to parallelize the algorithm operations. Master-slave GA is a basic approach to run GA in parallel. In this model, the main GA operations are done by a process called master. The master maintains the whole population and performs crossover, mutation, and selection operations. The rest of the processing units, which called workers or slaves; perform the evaluation of solutions only. The master-slave model is similar to the basic GA in behavior, except the parallel evaluation for the individuals. The main aim of this design is reducing the computing effort and hence reduce the execution time as illustrated in Figure 2.1.

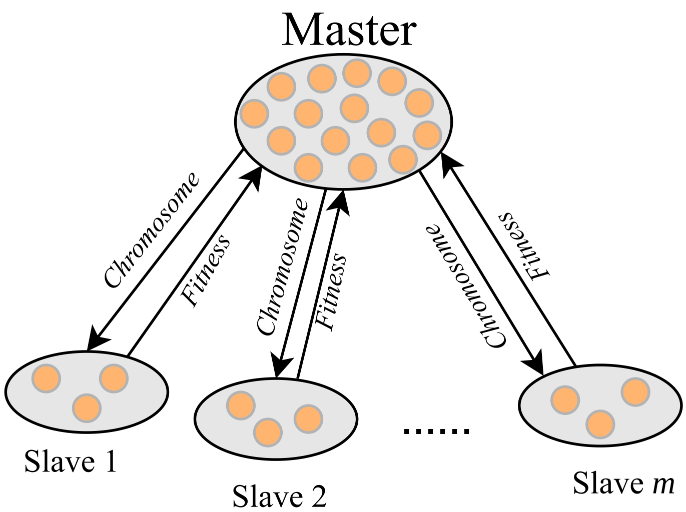


Figure 2.1: Illustration of master-slave model

Using this centralized communication type makes the master-slave model more efficient in shared-memory computing systems when the fitness function is complex and time-consuming [8]. Previous researches stated that master-slave models have the same search behavior and numerical effort of the serial GA [25, 50]. Early studies on designing efficient master-slave PGAs were presented by Cantu-Paz [25, 27]. He showed that adding more slaves to the execution pool will desirably decrease the computation time, but the communications time will increase absolutely. Thus, he developed the p-sets-p-slaves policy. That policy forces a trade-off makes the master-slave model does not scale-up well when more processing units are added to the execution pool over different hardware platforms.

UNIVERSIDAD DE MÁLAGA

Recent research strongly related to the previous one was proposed in [40]. In that research, a new mathematical model for measuring the performance of the master-slave paradigm EC techniques was proposed. Their results demonstrate that the p-sets-p-slaves policy developed by Cantu-Paz is not optimal. They proved that the number of communication cycles should be proportional to both the number of slaves and the communication time. They came up with another important conclusion that the master-slave architectures could reliably scale well over local area networks of workstations.

Another effort to study the performance of the master-slave model for solving the multi-objective optimization algorithms was presented in [85]. They studied the parallelization of multi-objective optimization algorithms using the master-slave model on a heterogeneous set of processors. In a heterogeneous environment, the waiting time for receiving solutions is high due to the different clock speed of the processors. Ignoring the high communication overhead between the processors, they aimed at solving very expensive optimization problems. Their algorithm tends to utilize the different computing resources (from the slow to the very fast processors). Their proposed approach could be suitable for very expensive multi-objective problems of real-world applications.

2.1.2 Island Model

Population-based metaheuristics maintain a population of n individuals exploring the problem space concurrently from different points of exploration. One of the most common parallel implementations is the island model, which thoroughly described on [8]. The population of solutions is structured into small sub-populations relatively isolated each from other. These sub-populations could evolve in physical parallelism on different processors or run on a single core (design and implementation are two different orthogonal domains). After a predefined period of running time, some selected individuals (or other information) are exchanged via a migration process. The migration of individuals allows the merging of the genetic material between the sub-populations, thus increasing diversity. Figure 2.2 shows an illustration of the island Model.

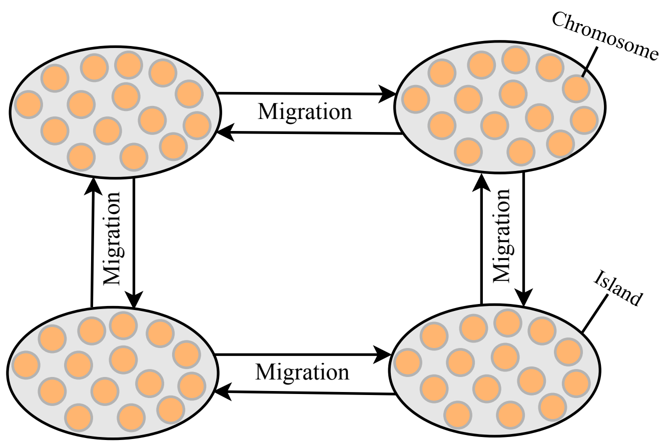


Figure 2.2: Illustration of the island model

UNIVERSIDAD DE MÁLAGA

The effect of synchronism of communications between these islands is crucial in the performance of the algorithm. We review some of the efforts in studying the effect of Synchronization of Communications on the performance of metaheuristics in the following section.

2.2 Parallel Performance: Synchronization

The numerical performance and parallel efficiency are highly correlated with the communication schemes used. In the past two decades, the researchers have done many studies on the effect of synchronization on parallel metaheuristics. One of the first attempts to analyze the performance of sync and async dGAs was presented in [14]. The authors analyzed the behavior of synchronism and migration over dGAs for eight processors. They presented a detailed comparison between the two implementations from a run-time point of view. They reported that synchronism does not numerically affect the GA when running in a homogeneous computing platform. For execution times, async algorithms outperformed their sync versions with a significant margin. The main outcome of their results is that the search behavior of the dGAs is highly affected by the communication scheme and migration operator configurations.

On another work, Alba et al. [11] presented the parallel version of a gradual distributed real-coded GA (GD-RCGA). They proposed a new parallel model named Hy3. Their work showed a comparative study of the effect of the synchronism on the behavior of Hy3 model for one and eight cores only. Regardless of the limited number of CPUs used in their study; their results are still a good reference to the difference between the two versions. Numerically both versions needed a similar effort for solving the problems, thus causing similar behavior for the sync and async migrations. For execution time, in the single CPU scenario, Hy3 sync model is faster than the async Hy3. However, Hy3 sync model is slower than Hy3 async model for eight CPUs experiment. Their main conclusion was that the async parallelization scales better than the sync one. Another effort to study the effect of synchronism on the PGAs behavior was presented in [75]. In that work, the authors presented a scalability analysis of the sync and async runs for solving a specific problem: the generalized assignment problem. Their experiments were conducted for large instances on multiple supercomputers. They developed an async migration operator to overcome the need for a global barrier effect in the communication between nodes. Their algorithm (named PGAP) was implemented in C, using MPI as a message-passing programming model. PGAP obtained better speedups (super-linear speedups observed) and was capable of exploring the solution space efficiently compared to the sync implementation for the considered problem. However, since they only used a single problem, more studies are needed to validate their conclusions.

Other attempts to study the synchronism of PEAs was presented in [98]. The authors proposed a study of the effect of the synchronism on the numerical performance of a parallel particle swarm optimization (PSO) on large-scale optimization problems. They addressed three different types of synchronism and presented a comparison between them. Their results show that the async PSO had the best general performance in large neighborhoods, while the sync PSO had the best one in small neighborhoods. Venter et al. [119] also studied the behavior of the sync and async PSO implementations. They concluded that the async implementation of the algorithm significantly outperformed the sync one in terms of parallel efficiency. For real-world problems, Serani et al. [110] studied the parameter selection in sync and async PSO setting for ship hydrodynamics problems. They presented a comparison of sync and async implementations under a set of limited resources.

UNIVERSIDAD DE MÁLAGA

2.2.1 Parallel Single-solution based Metaheuristics

Single-solution based metaheuristics proved to be effective optimization techniques. Many researchers have used the parallelism to improve their search accuracy and reduce execution time. For example, one of the recent proposals to study the parallel behavior of simulated annealing was proposed in [48]. The authors of the work implemented the parallel sync and async versions of the coupled simulated annealing (CSA) algorithm on a shared-memory architecture. The async version of the CSA obtained better solution quality and speed-up compared to the serial and to the sync one. In the area of parallel Evolutionary Algorithms (PEAs), there were some attempts to design and propose algorithms with new async schemes. For example, Harada et al. [55] proposed a parallel async EA with different synchronicity types for optimization problems. Also Santander-Jiménez et al. conducted in [109] a study of the async model of multi-objective optimization. Their model aimed at solving the performance drawback in the design of the algorithm by avoiding synchronization points, to improve the exploitation of parallel resources. They remarked that the sync implementation suffered from noticeable overhead penalties. As again expected, the async design succeeded in reducing the overhead impact and attained a statistically significant improvement in speedups over the sync approach.

For enhancing the search behavior of the sequential execution, many parallel approaches were proposed for these algorithms. Herrán et al. [56] proposed a parallel General VNS algorithm for solving the Hamiltonian p-Median Problem (HpMP), which is a generalization of the Traveling Salesman Problem (TSP). They employed several migration strategies by considering multi-start and parallelization schemes for improving the resulting algorithm. A comparison with the state-of-the-art techniques over a wide range of instances showed that their algorithm achieved the best quality with a shorter execution time in all instances. Another effort for parallelizing a single-solution based algorithm was proposed in [72]. In that study, a parallel SA was hybridized with a greedy algorithm with memoization to accelerate the search process. The resulting algorithm targeted improving the quality of the learned Bayesian network structures. Enhanced with the parallel search and the effectiveness of a more exhaustive search, the resulting algorithm showed better performance in terms of computational time and accuracy. Find more studies on parallelizing single-solution based metaheuristics on [86, 102].

2.2.2 Parallel Performance: Theoretical Analyses

Recently, there are advances in studying parallel optimization performance approaches theoretically. The Pareto Optimization for Subset Selection (POSS) method is an efficient approach for solving the subset selection problem. The authors of [96] proposed PPOSS, a parallel approach of POSS, and performed a theoretical analysis for its parallel efficiency. Their analysis shows that PPOSS has acceptable parallelization characteristics while preserving approximation quality for solving the problem. They proved that: under a limited number of processors (less than the number of variables), the running time of PPOSS can be reduced almost linearly. While with an increasing number of processors, the running time can be highly reduced (super-linear speed-ups).

UNIVERSIDAD DE MÁLAGA

Moreover, they proved the effectiveness and reliability of the asynchronous implementation of PPOSS. Another theoretical parallel optimization research approach correlated to the former theoretical approach is found in [120]. In that effort, the authors developed a theory to determine the timing and manner for scaling resource (up/down) for cloud-based MapReduce. MapReduce is a modern computation java model for parallel/distributed processing of big-data on clusters. They performed the theoretical analysis by applying a nonlinear transformation to define the problem in reverse resource space. The obtained theoretical results were employed in three theorems for guaranteeing the Quality of Service (QoS) of cloud-based MapReduce. For more theoretical investigations to the performance of parallel and distributed optimization approaches, we refer the reader to [8, 34, 90].

2.3 Parallel Metaheuristics: New Computing Platforms

Recently, there were some advances in studying the parallel metaheuristics in the cloud platforms, as new computing systems. Zhan et al. [124] proposed a new Differential Evolution (DE) algorithm for cloud platforms. The authors of that work aimed to design a new double-layered heterogeneous DE taking into account the challenges of *ad hoc* configurations of operators and parameters and the expensive run-time for real-world problems that face the basic DE. The first computing layer consists of evaluating a set of different concurrent populations with several parameters or operators. The second cloud computing layer consists of a set of cloud virtual machines evaluate the fitness of the populations in parallel to reduce the execution times, where the populations can be heterogeneous. In addition to the effectiveness of the new algorithm, the distributed cloud version of their algorithm achieved competitive speedups on expensive problems, compared to many other distributed DE and EC algorithms. For more efforts on designing and presenting new algorithms for solving complex optimization problems in the cloud platform, we refer the reader to [114, 117, 121].

A new effort for studying parallel model efficiency of distributed algorithms in multiprocessors was proposed in [3]. The authors of that effort investigate the performance of the distributed algorithms over different parallel models and communications schemes. They analyzed the performance of the master/slave, along with the distributed synchronous/asynchronous GAs models. Their experiments run over modern shared-memory multiprocessor system. They proved the efficiency of the asynchronous implementation of the distributed GA (dGA) over the synchronous one. Also, they proved that the master-slave parallel model was able to scale-up well over multiprocessors. Hofmann et al. [59] presented an interesting study on the performance of the parallel GAs on GPUs. Their main goal was investigating the suitability of GAs to be run on modern GPUs. Another goal of that study was examining which task and operators of GAs should be executed in parallel. They proved the efficiency of the parallelism of the population-based approach over GPUs. For more reliable performance, they concluded considering running all parts of the algorithm on the GPU. For more recent approaches on applying evolutionary algorithms on GPUs, we refer the readers to [31, 76, 88].

UNIVERSIDAD DE MÁLAGA

2.3.1 Parallel Metaheuristics: Surveys and Comparative Studies

In the literature, several surveys and comparative studies have been performed for different optimization algorithms and paradigms. The importance of comparative studies lays on revealing the characteristics of the algorithms under comparison. A new comparative analysis of two parallelization approaches for multi-objective optimization was proposed in [123]. In that study, two asynchronous parallelization variants for a multi-objective coevolutionary solver were designed and analyzed. The authors of that work stated that both parallelization approaches were able to show competitive convergence behavior of the baseline coevolutionary solver. A recent comprehensive survey on multi-population optimization algorithms was proposed in [77]. That study summarized and reviewed the up-to-date efforts on multi-population methods and its applications, considering the parallel approaches proposed for such algorithms. In consequence, they presented extensive surveying for the parallel algorithms and models applied for these algorithms, over CPU, parallel GPU, and multi-core architectures. On new generation metaheuristics algorithms, Dokeroglu et al. [38] presented a comprehensive survey of recent metaheuristics. They overviewed fourteen different algorithms, in addition to presenting the parallel approaches and paradigms employed for efficient performance. Del Ser et al. [37] presented a recent survey study on numerical optimization using the bio-inspired and evolutionary computing methods. The significance of their research lays in over-viewing the recent history of the bio-inspired computation and EC with a prospective look to the future of these methods. For more studies survey the parallel approaches of metaheuristics and evolutionary algorithms, we refer the readers to [70, 80, 108].

2.4 Energy Consumption of Parallel Metaheuristics

Overall, the literature on analyzing energy consumption of GAs exists, though it is very limited yet. As to PGAs, the literature is still shorter. We can also find several articles on nearby fields, like applying metaheuristics for problems dealing with energy minimization in a target scenario. However, that approach is different from our goal since they are not targeting the energy consumption of the algorithm itself but using the algorithm to solve problems concerns energy.

In the area of GAs, there are efforts to study the energy consumption behavior of the algorithm. One of these efforts was proposed by [41]. There, authors developed a GA to solve an extended version of the job-shop scheduling problem by considering its energy consumption. Another related effort was performed in [53], where authors evaluated the performance of three metaheuristic algorithms on the basis of cost and minimizing energy reductions. In the area of parallel algorithms, the authors of [81] proposed a parallel bi-objective hybrid genetic algorithm that takes into account energy consumption. They studied island and multi-start parallel GA models with a hybrid approach between a multi-objective PGA and energy-conscious scheduling heuristic. They concluded that the hybrid approach consumes more resources than the energy-aware scheduling heuristic, and the insular approach consumes more resources than the hybrid approach.

Previously mentioned efforts studied GAs and other algorithms to improve energy efficiency when solving a problem. However, they did not analyze the energy consumed by the GA algorithm or its components. In the scope of analysis of energy consumption of evolutionary algorithms (EAs), Vega et al. [36] presented a preliminary study on the energy consumption of the Genetic

UNIVERSIDAD
DE MÁLAGA

Programming (GP) algorithm. They run their experiments on different hardware devices over a number of operating systems. The main goal of their study was to show the effect of the main parameters of the algorithm on the energy consumption. They concluded that: devices with better processors can run the algorithm faster but spent larger amounts of energy. They also reported the influence of changing population sizes in the variable amount of energy required to reach solutions. Another recent research in this regard was proposed by Alvarez et al. in [16]. The authors of that work presented a preliminary energy consumption estimation model, based on the analysis of the influence of GP parameters on their energy consumption under a number of hardware devices. They concluded that their model was able to correctly estimate the energy consumption of the GP algorithm over the different devices.

We now turn to review the relevant researches concerns tools to measure energy consumption, either in algorithms or other types of software. Since it was proposed by Intel, Intel's Running Average Power Limit (RAPL) [35] interface has been used widely to measure the energy consumption of algorithms. In [24], authors used RAPL as a measurement to the execution of the algorithms and stated that it is reliable in many different types of computing systems. In another aspect, the authors of [118] used the RAPL interface to measure the energy consumption characteristics of MPI calls. They proposed a model to accurately measure the aggregate energy consumed by all processes engaged in MPI operations. In [105], authors extended Flex-MPI [78] with energy-aware and power-aware capabilities. Their aim was to increase the energy efficiency of parallel applications by means of malleability. All the power measurements in their novel approach were obtained by means of the RAPL interface. With a different objective, the work presented in [99] focused on the comparison between RAPL and two other power-meters methods for gathering energy consumption values. Their contribution was to study the correspondence or difference of the energy data provided by these methods. Their analyses show that RAPL has good correspondence due to using the faster and reliable hardware counters, which allows to use it for measuring energy consumption accurately. For more researches on using RAPL in energy measurement, we refer the reader to [19, 65, 93]. Thus, we could claim that RAPL proved to be a reliable tool for measuring the energy of algorithms.

In another recent research published in [29], a collaborative optimization algorithm for energy-efficient distributed flow-shop scheduling problem is proposed. The authors of that work aimed at solving the no-idle flow-shop scheduling effectively, targeting minimize the makespan and total energy consumption. They applied two heuristics collaboratively for population initialization, sponsored by multiple search operators for enhancing the exploration process. Furthermore, they designed different intensification strategies for enhancing the exploitation process. They stated that their algorithm improved the energy consumption by utilizing a speed adjusting strategy and was effective compared to other optimization algorithms used there. For more research efforts on designing energy-efficient evolutionary algorithms, we refer the readers to [49, 73, 112].

2.5 Summary

In summary, studying the parallel performance and the effect of synchronism on the behavior of PEAs have gained the attention of the researchers in the past three decades. Sync PEAs are a direct approach of parallelization, nevertheless, it suffers from overhead problems. Async approaches

UNIVERSIDAD DE MÁLAGA

come as the promising alternative of parallelization since it obtains better execution times and speed-ups compared to the sync one. Even if sync algorithms may have some performance issues, but in some specific problem or algorithm settlements it may come as a good and predictable design choice. Therefore, studying the effect of synchronism on the parallel and distributed algorithms, along with the energy consumption of the algorithms is an important issue both for scientific and real-world problems. In conclusion, we present a classification to the major related works of the literature in the areas of parallel performance and energy consumption analyses in Table 2.1.

There are several differences between the works listed in Table 2.1 and the proposal in this thesis. As we can see from the Table that most of the works ignored or neglected the energy consumption of the running algorithms. On the other hand, there are almost no research efforts were done to study the performance of the different parallel models linked with energy consumption. Here in the thesis, we want to fill this big gap in the area of parallel performance with experimental analysis to the energy consumption and run time. Another important issue is that most of the studies in the literature have focused on the typical computing platform (Cluster of computers), however, we targeted a new computing platform (Multiprocessors). Our approach here is to present a comprehensive study to the performance of sequential and distributed metaheuristics running on a homogeneous multiprocessor computing system, visiting their energy consumption profiles.

The previous works listed in Table 2.1 were dealing with either an application where energy was considered (out of the algorithm itself) or tools for measuring energy in software packages in general. The actual situation in this domain is that of a definite shortage of works focusing on GAs (a really important kind of techniques today) with modern tools for measurement. Besides, GA and PGAs are structurally different in many aspects, so they need and deserve a focused study, as we do here. Moreover, we objective to analyze the performance of three canonical metaheuristics (GA, SA, and VNS) and their distributed counterparts in a comparative analysis study. In a new different aspect, our experiments concentrate on showing the parallel-execution combinatorics with optimization concerning its energy consumption behavior. We also extend our analyses for a varying number of cores from one to 32, which will add scalability findings to the study of performance. We also present a new detailed analysis of the effect of the synchronism on the solution quality and parallel efficiency of the dGA. In short, we aim to establish the bases for general knowledge for designing efficient optimization algorithms in modern multi-core machines.

UNIVERSIDAD DE MÁLAGA

Table 2.1: Classification of the main related works, in chronological order

Authors	Year	Algorithm	Platform-used	Parallelism model	Energy analysis
Peterson et al. [94]	1990	GA/SA	Simulation/PC	Low level	NO
Alba et al. [14]	2001	GA	Cluster	Sync/Async	NO
Alba et al. [11]	2004	GA	Cluster	Sync/Async	NO
Dubreuil et al. [40]	2006	EC	Cluster	Master/Slave	NO
Venter et al. [119]	2006	PSO	Cluster	Sync/Async	NO
Tantar et al. [113]	2007	SA	Grid	Async	NO
Borovska et al. [21]	2007	SA	Multi-computer	SPMD	NO
Mostaghim et al. [85]	2008	Hybrid EA	Cluster	Master/Slave	NO
Mezmaz et al. [81]	2011	Hybrid GA	Grid of clusters	Island model/async	Yes
Hofmann et al. [59]	2013	GA	GPU/Multiprocessors	Island model	NO
Venkatesh et al. [118]	2013	MPI primitives	Multi-core cluster	N/A	Yes
Iturriaga et al. [62]	2013	Local search	Cluster	Multithreading	Yes
Xu et al. [121]	2014	Cuckoo search	Cloud	Async	NO
Dorronsoro et al. [39]	2014	Heuristic	Multi-cores	High level	Yes
Liu et al. [75]	2015	GA	Cluster	Sync/Async	NO
Zhan et al. [124]	2016	DE	Cloud	Master/Slave	NO
Escamilla et al. [41]	2016	GA	Machine/diff. speeds	N/A	Yes
de Vega et al. [36]	2016	GP	Diff. machines	N/A	Yes
Roberge et al. [104]	2016	PSO	GPU	Sync	Yes
Guzman et al. [53]	2017	3 Metaheuristics	PC	N/A	Yes
Santander-Jiménez et al. [109]	2017	EA	Multi-core cluster	Async	NO
Harada et al. [55]	2017	EA	Cluster	Diff. Sync. types	NO
Álvarez et al. [16]	2017	GP	Diff. machines	N/A	Yes
Coelho et al. [32]	2017	Metaheuristic	GPU	Threads	No
Gonçalves-e-Silva et al. [48]	2018	SA	Cluster	Sync/Async	NO
Munguía et al. [86]	2018	Neighb. search	Cluster	Sync	NO
Tsai et al. [117]	2018	Metaheuristics framework	Cloud	Sync	NO
Rios et al. [102]	2018	Neigh. search	GPU/Multi-core	Sync/Async	NO
Teijeiro et al. [114]	2018	DE	Cloud	Islands model	NO
Herrán et al. [56]	2019	VNS	Cluster	Diff. Parall. Schem.	NO
Lee et al. [72]	2019	Hybrid SA	Cluster	Sync/Async	NO
Cheng et al. [31]	2019	GA	GPU	Sync	Yes
Luo et al. [76]	2019	GA	GPU	Hybrid framework	Yes
Nitisiri et al. [88]	2019	GA	GPU	Low level/threads	No
Gong et al. [49]	2020	Hybrid VNS	PC	N/A	Yes

UNIVERSIDAD
DE MÁLAGA

Chapter 3

Summary of Results

In this chapter, we present a summary of the results of publications and original contributions of this thesis. These results and experiments were done to achieve the objectives and goals of the thesis. Here, we present in brief detail these publications, the full results and discussions could be found in the corresponding publishing-house.

We have five contributions, three of them published in JCR journals and two in international conferences. These publications are presented in the following order:

[1] Amr Abdelhafez and Enrique Alba. **Speed-up of synchronous and asynchronous distributed genetic algorithms: A first common approach on multiprocessors.** 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2677-2682, 2017.

Swarm and Evolutionary Computation: Q1, Impact factor: **6.33**.
[2] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors.** Swarm and Evolutionary Computation, 49:147-157, 2019.

The Journal of Supercomputing: Q2, Impact factor: **2.16**.
[3] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **A component-based study of energy consumption for sequential and parallel genetic algorithms.** The Journal of Supercomputing, 75(10):6194-6219, 2019.

[4] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Analyzing the energy consumption of sequential and parallel metaheuristics.** 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019.

Swarm and Evolutionary Computation: Q1, Impact factor: **6.33**.
[5] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Parallel Execution Combinatorics with Metaheuristics: Comparative Study.** Swarm and Evolutionary Computation, 2020.
DOI: https://doi.org/10.1016/j.swevo.2020.100692

The full list of the publications and a description for the corresponding journal or conference can be found in Appendix B.

UNIVERSIDAD
DE MÁLAGA

3.1 Speed-up of Synchronous and Asynchronous Distributed Genetic Algorithms: A First Common Approach on Multiprocessors

Amr Abdelhafez, and Enrique Alba
IEEE Congress on Evolutionary Compu. (CEC), pp. 2677-2682, 2017

GAs are being used to solve a wide range of real-world problems, thus it is important to study their implementations to improve the solution quality and reduce the execution time. This article presents a basic study of the speed-up of PGAs, where a common approach is followed for better understand synchronous and asynchronous algorithms. We analyze the behavior of GAs over a homogeneous multiprocessor system. We report results showing linear and even super-linear speed-up in both cases of study.

GAs have weak features like slow convergence and time-consuming processes, especially when being applied to a problem with many parameters and enormous search space. It may take weeks and even months to accomplish an optimization task. GAs can readily be parallelized in the form of islands of subpopulations. These subpopulations can collaborate by running them on separate processors: The so-called dGA. A new behavior then emerges because of the different processes work together to find a solution to the same problem. The parallel communication scheme is very important in determining the convergence behavior of the algorithm [12]. In the synchronous algorithm, there is a synchronization point, which means that all the islands should move from one communication stage to another together. The asynchronous parallel algorithm does not have any synchronization points between islands, thus islands proceed on their own with sparse and non-synchronous exchanges of information. We wonder what are the resulting speed-up and numerical behavior of a dGA of these two kinds when running in a multiprocessor system.

The scientific contribution of the work lies in the combined study of synchronous and asynchronous dGAs from a run time and a numerical point of view over multiprocessors. Contrary to most other studies we run the algorithms on a multiprocessor system, that can potentially vanish the numerical and run times differences (or not). Since the aim of this study is the parallel algorithm and its numerical behavior in a multiprocessor system, we use the common and well-known one-max problem for this first study. Our migration topology is based on the uni-directional ring topology described in [107], where it was ranked as a good way to go out of the 14 different migration topologies studied there. In our experiments, we address both synchronous and asynchronous implementations using the same parameters for the same problem. We perform our experiments in a homogeneous dedicated multiprocessor without interruptions from any other processes.

Tables 3.1 and 3.2 present the results of the synchronous and asynchronous implementations, respectively. These results are the average number of evaluations required in the 30 independent runs for finding the optimal solution. All the runs considered problem dimensions of 1000, 2000, and 3000 for eight islands, and had a 100% success rate. The obtained results show that our algorithm achieved consistently excellent results in all the tested dimensions in both cases of the study. We can see in fact that the numerical effort was similar in all cases between the sync and async algorithms: that is common sense, since the base algorithm is the same (always using 8 islands

UNIVERSIDAD DE MÁLAGA

Table 3.1: Mean number of evaluations of the synchronous dGA

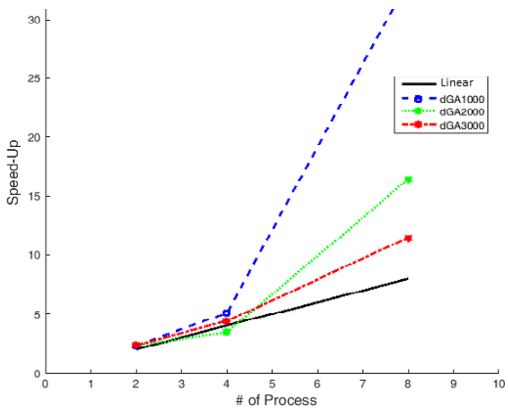
$No.\ of\ CPUs$	$l = 1000$	$l = 2000$	$l = 3000$
1	7,556,507	17,534,680	28,383,987
2	7,439,840	17,791,120	27,832,160
4	7,701,000	17,276,506	28,332,920
8	7,420,413	17,446,026	27,947,467

Table 3.2: Mean number of evaluations of the asynchronous dGA

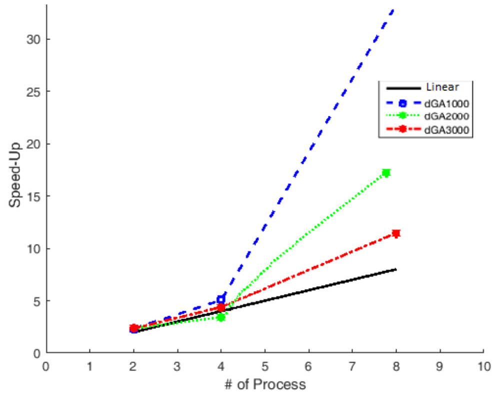
$No.\ of\ CPUs$	$l = 1000$	$l = 2000$	$l = 3000$
1	7,513,003	17,255,383	28,460,233
2	7,411,627	17,750,387	28,094,413
4	7,562,707	17,578,880	28,200,040
8	7,440,907	17,767,560	28,198,733

over any of the numbers of processes), and then only random fluctuations can be expected. This behaviour was expected and now it is confirmed by real experimentation.

Figure 3.1 shows the speed-up of our algorithms on dimensions 1000, 2000, and 3000. The figure clearly shows that our techniques constantly achieve a good speed-up in all of the studied dimensions for a lower number of cores used in this experiment.



(a) Synchronous model



(b) Asynchronous model

Figure 3.1: Speed-up of the sync/async dGA schemes

These results demonstrate that the two implementations are equally effective and fast from a numerical effort perspective, which is not always the case in a cluster of computers [14]. In both synchronous and asynchronous cases, our algorithm obtained high efficiency in finding the optimal solution for high dimensional problems. The parallel performance of the synchronous and asynchronous versions is very good in a multiprocessor computer, both in terms of time and solution quality. We finally discovered that speed-up is very high and even super-linear in multiprocessors. The use of a different problem will of course vary the results, but GAs and metaheuristics are very especial algorithms since the probability of getting a solution at any time is non-zero. Thus, parallelism is having a high chance of getting very fast convergence to the optimal solutions. For complex search spaces, this can relate in unexpected ways with the use of more processors: more memory (and cache) and the work of operators can yield super-linear results [7].

UNIVERSIDAD DE MÁLAGA

3.2 Performance Analysis of Synchronous and Asynchronous Distributed Genetic Algorithms on Multiprocessors

Amr Abdelhafez, Enrique Alba, and Gabriel Luque
Swarm and Evolutionary Computation, 49:147-157, 2019

This article is an extension of the research described in the previous section, where preliminary results of the island dGA were obtained for one problem (ONEMAX) for eight cores only. In this research, we extend our preliminary study by analyzing new problems (ONEMAX, MTTP, ECC, and P-PEAKS). These problems possess different search spaces and different dimension sizes. First, we analyze one of these problems with several dimension sizes. Second, our experiments consider a higher number of cores, from one to 32 cores. Thus, we could show the effect of the synchronism on the distributed algorithms, using a different number of problems and dimensions over a various number of cores. Moreover, we present the numerical results and speed-up of the master-slave model. The scientific contribution of this paper lies in the combined study of the different parallel models for GAs from a run-time and a numerical point of view. We aim at allowing researchers and final users to build new efficient search techniques that fit their goals.

Tables 3.3 and 3.4 present the numerical effort (number of function evaluations) needed for the master-slave, sync, and async dGA implementations to find the optimal solution.

Table 3.3: Mean number of function evaluations and standard deviation of master-slave GA

Problems	ONEMAX		MTTP20		MTTP100		ECC		P-PEAKS	
# of cores	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	101509.50	35463.01	4108.53	804.80	927030.24	127400.84	28272.00	14914.80	3583.60	244.18
2	98105.36	2950.71	4016.22	778.62	807581.30	325641.23	24180.00	4705.88	3485.45	366.50
4	89974.40	24535.80	3892.22	819.37	775372.74	1010187.32	29784.80	23186.57	3509.20	165.85
8	90548.62	13583.53	3893.60	834.63	741272.00	243470.19	31723.33	18626.91	3596.95	248.37
16	81319.20	19219.58	4176.84	849.15	882384.02	631929.40	20184.00	6489.51	3583.60	237.08
32	128991.00	31371.46	3611.50	786.69	970533.00	2015475.74	22989.60	2282.41	3757.33	657.29

The obtained results show that these algorithms achieved consistently excellent results in all the tested dimensions in all problems. Table 3.3 presents the results of the master-slave model for five instances from our study. We can observe that the differences in the number of function evaluations when using a different number of cores are not high. This is an excepted result since the behavior of this technique does not change with the number of slaves: its numerical behavior is the same as the serial version but in a faster way. In summary, the obtained results demonstrate the reliable search capability of the master-slave model in solving different search problems over multiprocessors.

From the results presented in Table 3.4, we can see in fact that the numerical effort of finding the optimal solution was reasonable in most of the cases for the sync and async cases respectively: that is common sense since the base algorithm is exactly the same (always using 32 islands over a varying number of cores). In our analysis, we should recall that finding a solution in GAs at any generation is predictable. The async version is using a fast non-blocking communication scheme, but it may result in spending more generations in finding a solution with the same quality as the

UNIVERSIDAD DE MÁLAGA

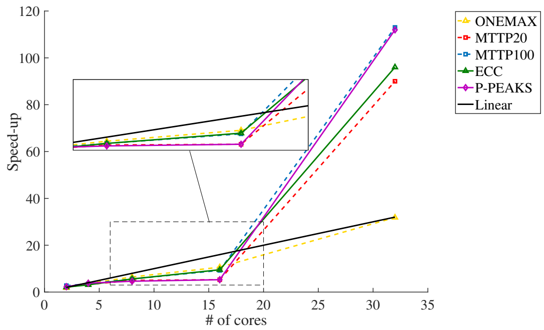
UNIVERSIDAD DE MÁLAGA

Table 3.4: Mean number of function evaluations and standard deviation of sync/async dGAs

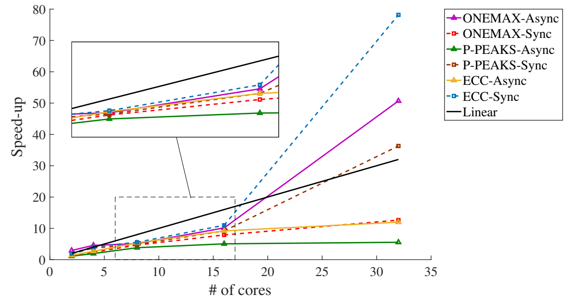
Problems		ONEMAX		MTTP20		MTTP100		MTTP200		ECC		P-PEAKS	
# of cores	Metric	Sync	Async	Sync	Async	Sync	Async	Sync	Async	Sync	Async	Sync	Async
1	Mean	**273731.67**	304421.67	191113.33	**177803.33**	**4729915.00**	6585970.00	9212008.33	**9196250.00**	203333.33	**199240.00**	**55026.67**	57863.33
	SD	3447.42	5409.06	83302.57	76311.61	1701614.66	1637011.23	674518.44	111580.74	23868.27	27311.78	2430.01	3589.61
2	Mean	**273165.00**	306633.33	188006.67	**174615.00**	**5204773.33**	5970097.92	9426566.67	**9077450.00**	**200276.67**	203090.00	**54745.00**	58251.67
	SD	2642.69	4817.63	70349.06	64488.11	1335904.38	1381881.51	459063.23	759081.79	23868.27	27311.78	2441.01	3499.45
4	Mean	**274356.67**	306935.00	204165.00	**191716.67**	**4941330.00**	6092321.43	**8479200.00**	9633800.00	**193981.67**	199581.67	**54156.67**	56206.67
	SD	3517.19	7236.45	76419.69	92678.19	1388687.00	1787543.26	611718.07	243527.57	23868.27	27311.78	3592.71	3748.42
8	Mean	**273310.00**	305926.67	174440.00	**156853.33**	**5009773.33**	6897930.43	**9129138.89**	9629525.00	203096.67	**182156.67**	55156.67	**33321.67**
	SD	3258.24	8713.61	69293.75	90047.59	1175499.01	1474699.95	652022.39	671221.78	23868.27	27311.78	3039.41	14745.38
16	Mean	**273136.67**	310153.33	172513.33	**135335.00**	**4588078.33**	5566586.54	8921507.14	**8237450.00**	203531.67	**163288.33**	55475.00	**31068.33**
	SD	2867.38	19764.68	65806.25	60148.99	1452820.03	1590190.31	931479.63	702793.42	23868.27	27311.78	3231.90	6065.67
32	Mean	**276650.00**	305766.67	177073.33	**172513.33**	**4775386.67**	5202036.21	**8616025.00**	9271743.75	201730.00	**196610.00**	59031.67	**46718.33**
	SD	2893.99	5480.87	54497.41	79995.96	1325606.84	1674027.40	745693.03	491609.53	23868.27	27311.78	3320.80	14051.10

We highlight the best values between each case in bold sync algorithm. Referring again to Table 3.4, in consideration to our experiment settings and the termination condition, we could determine that the difference in the numerical performance between the async and sync algorithms is not high in multiprocessors computing systems.

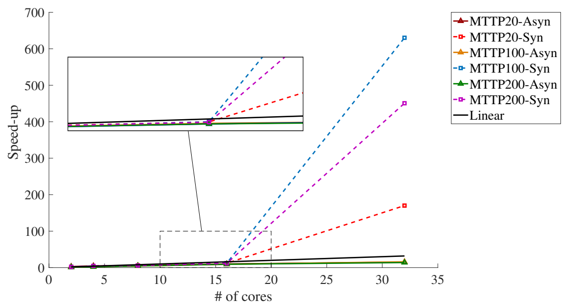
Figure 3.2 offers a graphical visualization of the results.



(a) Master-slave model



(b) Sync/async dGAs (ONEMAX, P-PEAKS, and ECC)



(c) Sync/async dGAs (MTTP20, MTTP100, and MTTP200)

Figure 3.2: Speed-up of different dGA implementations

From the experimental analysis, and now Figure 3.2(a) proves our hypothesis on the master-slave model: for all the problems under the study, the master-slave model succeeds in scoring a higher speed-ups constantly. Figures 3.2(b) and 3.2(c) support our discussion on the speed-up performance of the sync and async implementations. The sync algorithms (in dotted lines) has a different run time and thus speed-up behavior than the async ones (solid lines) over multiprocessors.

In summary, for all the dGA implementations under the study, using more cores resulted in better speed-ups, what is good for the expected scalability of these algorithms under multi-cores systems. The master-slave model showed competitive numerical effort with perspective with those of the island model dGA. The master-slave model proved to utilize efficiently new processing units resources that become available nowadays, whereas the effective design of the island-model can efficiently exploit the search space. We claim that the island-based GA is numerically efficient more than the master-slave model over multiprocessors, by assuring the integration of the decentralized design and migration operator.

UNIVERSIDAD DE MÁLAGA

3.3 A Component Based Study of Energy Consumption for Sequential and Parallel Genetic Algorithms

Amr Abdelhafez, Enrique Alba, and Gabriel Luque
The Journal of Supercomputing, 75(10):6194-6219, 2019

Recently, energy efficiency has gained attention from researchers interested in optimizing computing resources. Solving real-world problems using optimization techniques requires a large number of computing resources and time, consuming an enormous amount of energy. In particular, GAs are hardly found explained in their internal consumption behavior. In the present article, we analyze the energy consumption behavior of such techniques. We expand our study to include several algorithms and different problems and target the components of the algorithms so that the results are still more appealing for researchers in arbitrary domains of application. We focus on the measurement and quantitative analysis of the energy consumption of GAs and PGAs. We perform an extensive analysis of the energy consumption and execution time of the components of a sequential GA. Further, we include the energy consumption analysis of PGAs with two different communication schemes (synchronous and asynchronous).

The main contributions of this paper are to study the energy consumption and execution time behavior of both sequential and parallel GAs, in reasonably wide energy analysis of its components under different computer communication schemes. The potential impact of these results in building new techniques that fit the aims of green computing will also link to a line of research for making algorithms more efficient as a piece of software running on a computer. This point of view not so present in literature, but a fundamental one for formulating high-quality research.

In this experiment, we analyzed the energy consumption of the sequential panmictic GA. Table 3.5 and Figure 3.3 present the energy consumption percentages of the GA components, respectively. We separate problems of $O(n)$ and $O(n^2)$ in the table with a dashed line. In Table 3.5, we boldface the components with the highest energy consumption percentages for each problem, respectively.

Table 3.5: Energy consumption percentages (%) of the different GA components

Problem	Evaluation	Crossover	Mutation	Housekeeping
COUNTSAT	22.02	26.52	**26.87**	24.59
MTTP	17.62	31.30	**38.74**	12.34
FMS	**61.38**	14.86	20.64	3.12
ONEMAX	3.87	36.25	**50.45**	9.43
MAXCUT	**85.01**	5.53	7.25	2.22
P-PEAKS	**86.48**	5.66	7.71	0.15
ECC	**69.90**	11.58	16.22	2.31
MMDP	8.22	35.13	**49.15**	7.51

We can get several conclusions out of Table 3.5 and Figure 3.3. The evaluation operator consumes most of the energy in four of the problems, while for the other four problems, the mutation operator consumes most of the energy. It also confirms another important observation: that mutation represents a potential energy hotspot component inside the GA. The mutation consumption is, however, more subtle. This operator is constantly generating random numbers to decide whether a

UNIVERSIDAD DE MÁLAGA

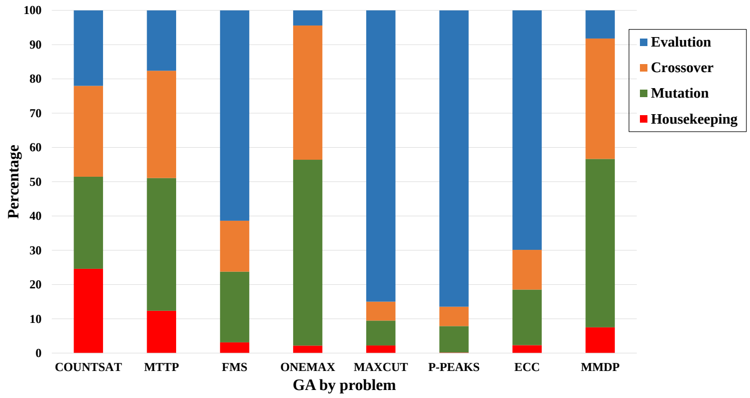


Figure 3.3: Energy consumption percentages (%) of GA components

bit should be flipped or not. The random number generation is shown to be quite an expensive operator (in time and energy) and the reason for the mutation consumption behavior. Housekeeping represents the smaller energy percentages in seven out of our eight problems: less than 15% of the total energy consumed by the algorithm. The only exception is COUNTSAT, where it goes up to 24.59% of the total energy consumption. This is due to the component has some operations which are independent of the features of the problem, and for easy problems (low dimensionality) with low energy consumption, this fix cost could be important when it is compared with the cost of the classical GA operations.

In another experiment, we present an analysis of synchronous and asynchronous dGA parallel models. Both implementations have the same parameters with the only difference being their communication scheme. Our algorithm consists of 32 islands for all the experiments done, and we study an increasing number of cores for running these 32 islands, from one to 32 cores. Table 3.6 present the results of energy consumption of the dGA implementations. We present the results for six problem instances: ONEMAX of size 2000 bits, P-PEAKS of 100 bits, ECC of size 288 bits, and three instances of MTTP named MTTP20, MTTP100, and MTTP200.

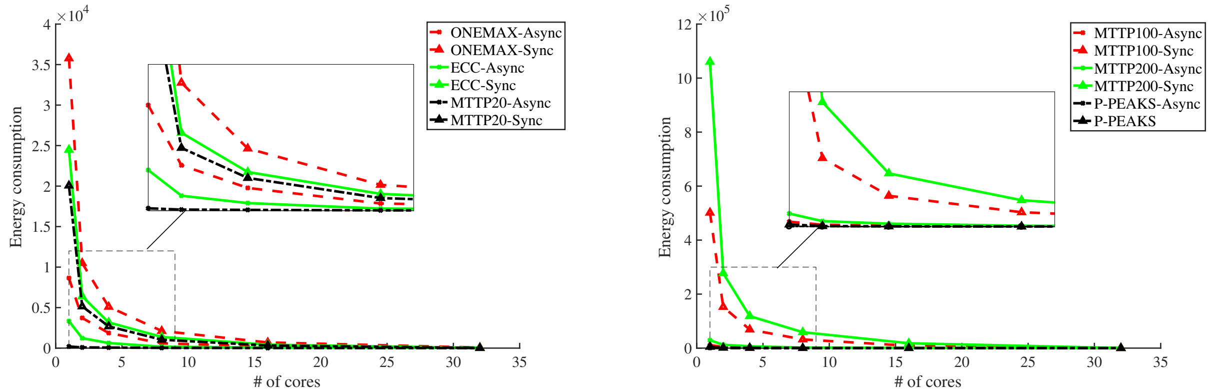
Table 3.6: Mean of energy consumption values on both versions, in kWh

Problem	ONEMAX		P-PEAKS		ECC		MTTP20		MTTP100		MTTP200	
# of cores	Sync	Async	Sync	Async	Sync	Async	Sync	Async	Sync	Async	Sync	Async
1	9.94E-03	**2.40E-03**	1.34E-03	**1.83E-04**	6.80E-03	**9.25E-04**	5.57E-03	**5.47E-05**	1.39E-01	**3.09E-03**	2.95E-01	**8.11E-03**
2	2.92E-03	**1.03E-03**	3.79E-04	**8.46E-05**	1.78E-03	**3.41E-04**	1.43E-03	**2.50E-05**	4.24E-02	**1.21E-03**	7.70E-02	**3.43E-03**
4	1.42E-03	**5.19E-04**	1.80E-04	**4.73E-05**	8.81E-04	**1.73E-04**	7.46E-04	**1.85E-05**	1.91E-02	**6.35E-04**	3.29E-02	**1.85E-03**
8	5.89E-04	**1.64E-04**	8.47E-05	**1.19E-05**	3.83E-04	**4.81E-05**	2.88E-04	**7.58E-06**	8.91E-03	**2.23E-04**	1.64E-02	**5.71E-04**
16	1.94E-04	**5.55E-05**	2.65E-05	**5.08E-06**	1.17E-04	**1.75E-05**	9.06E-05	**3.14E-06**	2.44E-03	**6.03E-05**	4.99E-03	**1.68E-04**
32	**1.95E-05**	2.06E-05	2.80E-06	**2.27E-06**	7.98E-06	**6.93E-06**	2.04E-06	**1.36E-06**	2.67E-05	**2.13E-05**	7.64E-05	**6.72E-05**

The results of Tables 3.6 clearly show that the asynchronous implementation consumes lower energy (marked in bold) in most of the instances of our experiments. These results prove that the asynchronous implementation is more energy-friendly than the synchronous implementation. Also, the larger the number of cores in the execution pool, the lower the energy consumption we have. Now we have quantitative evidence that parallelism can help reducing energy. Even though adding more computing units may theoretically speaking lead to consuming more energy, but adding more computing units steadily decrease the execution time and thus decrease the energy consumed (for the benchmark considered in these experiments).

UNIVERSIDAD DE MÁLAGA

Figure 3.4 shows the behavior of the energy consumption of synchronous and asynchronous algorithms respectively over a different number of cores.



(a) Problems: ONEMAX, ECC, and MTTP20 (b) Problems: MTTP100, MTTP200, and P-PEAKS

Figure 3.4: Energy consumption (in kWh) of the sync and async algorithms

The results of the Table 3.6, and Figure 3.4 clearly point to the fact that the asynchronous implementation is more efficient than the synchronous implementation: it consumes less energy and time in most of the cases of the study (for the same numerical performance). In terms of execution time, this behavior was reported in many previous studies on synchronism of EAs, e.g., [14]. Our results confirm this poor performance of the synchronous implementation in terms of energy consumption too. The reason for this behavior is the unblocking of communications of the asynchronous implementation, which means there are no idle cores waiting for incoming data. Furthermore, we presented a statistical comparison between asynchronous and synchronous implementations based on the energy consumption values obtained from both versions, by using the Wilcoxon's signed-ranks test. The results clearly show that both versions have a different energy consumption profile when being executed over a different number of cores. This outcome also proves our previous results in Table 3.6, so now we can claim that the asynchronous implementation has a different energy consumption behavior which is more efficient than the synchronous implementation.

In summary, our experiments on the sequential GAs show the controlling role of the fitness operator on energy consumption. It also reveals possible energy hotspots in GAs operations, such as the mutation operator. Mutation came out as the most consumption component in four of the problems of the study. Mutation scored higher energy consumption than crossover in all of the problems of the study. In this work, we also presented an analysis of the relationship between problem size and energy consumption. The analysis reveals that the energy consumption percentage consumed by GA operators is varying with the change of the dimension. Besides, our distributed evaluations besides a statistical analysis of the results demonstrate that the communication scheme could highly affect the energy consumption of the parallel evaluations of GAs. With respect to dGA, the results clearly point to the higher efficiency of the asynchronous version (time and energy), which we noticed for all numbers of cores. The statistical analysis did also confirm their different energy consumption profiles. We want to remark that the optimal energy consumption in the dGA configuration happened when using a number of islands equal to the number of cores.

UNIVERSIDAD DE MÁLAGA

3.4 Analyzing the Energy Consumption of Sequential and Parallel Metaheuristics

Amr Abdelhafez, Gabriel Luque, and Enrique Alba
Int. Conf. on High Performance Comput. & Simulation (HPCS), 2019

Nowadays, energy efficiency is taken into consideration during the design of new algorithms because of the million times that algorithms run on labs and computation centers. This work presents two novel experiments for investigating the numerical performance and energy efficiency of sequential and parallel metaheuristics. The main aim of this study is to analyze the energy consumption of three well-known and commonly-used metaheuristics: GA, VNS, and SA, and their parallel versions of 32 cores. We, therefore, propose an analysis of the performance of the sequential and distributed metaheuristics using the same accuracy goal and hardware equipment, software approach, and problem settings. In the literature, there are two main termination conditions used. The first condition is using the number of function evaluations as a termination condition. The second termination condition is arriving at a specific (optimal or good-enough) fitness-value.

We perform our experiments for the considered algorithms exhibiting two perspectives. The first experiment investigates the interconnection between energy consumption, execution time, and solution quality when a fixed number of evaluations used as a termination condition. In consequence, we perform a second experiment investigating the performance of the algorithms when arriving at a specific fitness-value as the termination condition. The contributions of this work are the investigation of the performance of six different algorithms, attending to their energy consumption profiles. We study the performance of the considered algorithms by using different benchmark and real-world problems. We consider a set of eight instances: three different cases of the multimodal problem generator (P-PEAKS) [63], and five instances of the Vehicle Routing Problem (VRP) [22].

Table 3.7 and Figure 3.5 present the average energy consumption values of the algorithms for the first experiment, in Joules. The terms dVNS, dSA, and dssGA refer to the distributed versions, respectively.

Table 3.7: Average energy consumption values of the algorithms, in Joules

Problem	VNS	SA	ssGA	dVNS	dSA	dssGA
P-PEAKS 20	517.08	762.11	785.18	59.81	**33.31**	39.23
P-PEAKS 50	2072.03	2097.82	1559.32	245.77	113.23	**64.38**
P-PEAKS 100	4234.43	4127.83	4130.03	454.49	202.32	**109.47**
vrpnc1	84.56	84.36	173.22	33.13	**19.40**	26.46
vrpnc2	110.24	109.24	278.23	32.40	**19.25**	25.10
vrpnc3	150.61	179.81	472.94	32.82	**19.12**	29.68
vrpnc4	258.72	257.96	813.90	37.25	**19.14**	30.43
vrpnc5	346.20	460.35	1232.90	33.87	**21.61**	34.90

Boldfaced and underlined represent the least and highest values, respectively

The values reported in Table 3.7 are the energy at the end of the execution. These values mainly depend on two factors: the number of algorithmic steps which can be calculated simultaneously

UNIVERSIDAD DE MÁLAGA

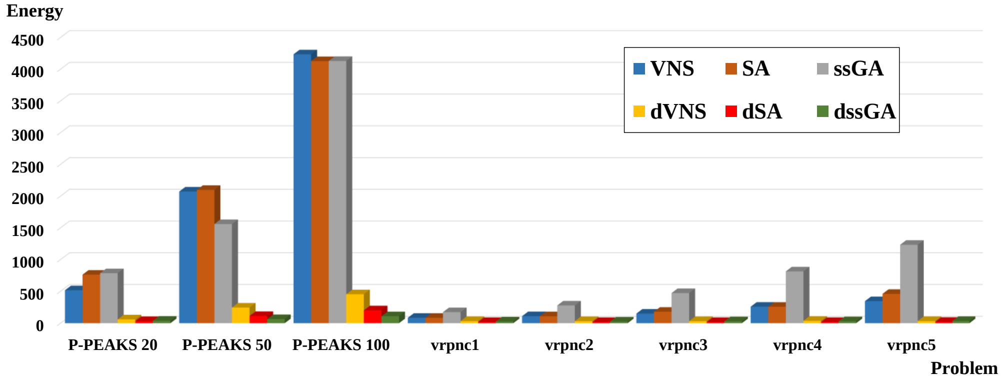


Figure 3.5: Energy consumption in Joules for the algorithms under the study (Exp.1)

(parallelism level) and the time needed to evaluate one step. We can observe that parallel algorithms consume a significantly lower amount of energy than the sequential ones. In concrete, dSA scores the least amount of energy consumption (in bold) in six out of the eight instances under study, while dssGA is the most energy-aware algorithm for the other two problem instances. The parallelism allows executing several algorithmic steps at the same time, thus reducing in a significant way the execution and consequently the energy consumed as shown in Figure 3.5. Using more cores is largely compensated by the reduction of the time of using these resources. The difference in the energy consumed by the algorithms depend on the special features of each one, such as crossover operator in ssGA or the calculation of the acceptance criterion in SA.

In the second experiment, we deploy another termination condition: finding a solution with a predefined quality for all the algorithms, with a max of 10^6 function evaluations. We determined these predefined values by a set of preliminary experiments, to get assured that all the algorithms would arrive at this target fitness. Table 3.8 and Figure 3.6 present the average energy consumption of the algorithms. The results are the average of the 30 independent runs, where all the algorithms were able to reach the optimal solution or the predefined fitness value.

Table 3.8: Average energy consumption values of the algorithms, in Joules

Problem	VNS	SA	ssGA	dVNS	dSA	dssGA
P-PEAKS 20	13.56	5.87	2.25	<u>37.62</u>	8.56	**2.16**
P-PEAKS 50	39.95	14.78	4.21	<u>72.10</u>	14.82	**3.70**
P-PEAKS 100	50.46	29.60	10.90	<u>134.43</u>	25.80	**6.29**
vrpnc1	1.42	**0.95**	9.05	<u>12.73</u>	6.80	2.37
vrpnc2	9.54	2.28	<u>31.66</u>	13.31	12.93	**2.01**
vrpnc3	2.14	2.80	<u>33.24</u>	6.14	8.13	**2.63**
vrpnc4	4.59	5.02	<u>81.00</u>	9.27	13.55	**2.98**
vrpnc5	**3.46**	3.75	<u>69.98</u>	5.67	10.95	3.47

Boldfaced and underlined values represent the least and highest values, respectively

Despite the distributed run on 32 cores, dssGA consumed the least amount of energy consumption (in bold) in six instances of our study. The distributed run and the cooperation between islands are the main keys in reducing the numerical effort, thus reducing the execution time and energy consumption of the search techniques. The ssGA uses population of solutions, thus a wider exploration of the search space is expected. Figure 3.6 clearly shows that: dVNS and VNS consumed

UNIVERSIDAD DE MÁLAGA

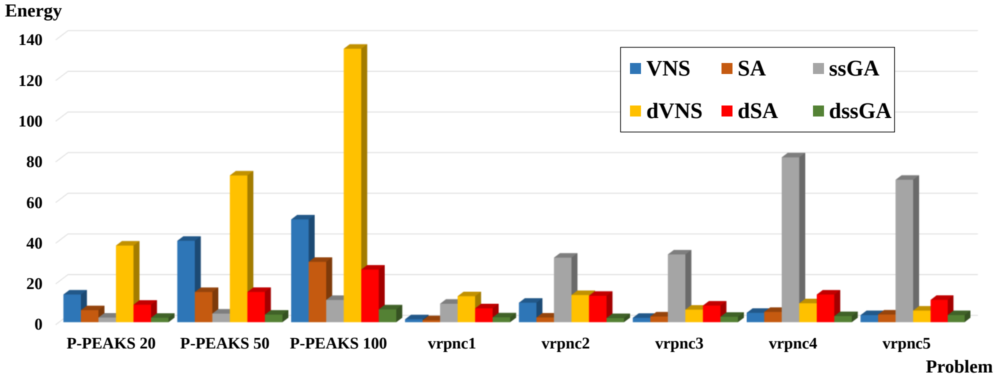


Figure 3.6: Energy consumption in Joules for the algorithms under the study (Exp.2)

the highest energy values for finding a solution for the P-PEAKS problem. P-PEAKS is a multi-modal function, so the basic VNS (without local search) requires a higher numerical effort to find the optimal solution. For the VRP, ssGA scored the highest energy consumption values in most of the cases. Using a sequential population of solutions in ssGA is the main key for this high energy consumption behavior.

The results and discussions of the different experiments show and confirm the interconnection between energy consumption, execution time, and solution quality for the considered algorithms. Under the same termination conditions, trajectory-solution based algorithms (VNS, SA) outperformed the population-based algorithms (ssGA) in terms of energy consumption and execution time and also the distributed versions outperformed the sequential versions in terms of energy consumption and execution times in most of the cases. Using the number of function evaluations as termination condition, the SA and its distributed version (dSA) consumed the least amount of time and energy in most of the cases among all of the other algorithms under study. When finding a solution with a predefined quality is the termination condition, dssGA outperforms the other algorithms in terms of time and energy consumption amounts. In this work, we also presented an extensive study to the numerical performance and execution times of the algorithms under study.

The parallel execution of the algorithms and knowledge sharing between the distributed algorithms proved to be a promising approach in reducing the energy consumption of search techniques. The overall outcome of the two experiments shows a trade-off between energy consumption and execution time from a side with numerical effort, and the number of function evaluations from another side. Using the number of function evaluations only as the termination condition to benchmark or comparing the algorithms is not fair and not suitable for the requirements of the present era. Such a condition would be misleading if used to compare the numerical performance of the different sequential and distributed algorithms. The discussions reveal the characteristics of the different sequential/parallel algorithms, which include trajectory-based and population-based metaheuristics so that this study is useful for the future design of energy-aware algorithms.

UNIVERSIDAD DE MÁLAGA

3.5 Parallel Execution Combinatorics with Metaheuristics: Comparative Study

Amr Abdelhafez, Gabriel Luque, and Enrique Alba
Swarm and Evolutionary Computation, 2020

In this work we present two extensive studies to the solution quality, energy consumption, and execution time for three different metaheuristics (GA, VNS, and SA) and their distributed counterparts. The main aim of our study is exploring the efficiency of parallel execution of the metaheuristics while being running in a multicore machine. This study differs from the studies explained in the previous sections in the termination criteria and the comparison examination between different algorithms. Here, we want to identify the combination and combinatorics between metaheuristics and solving optimization problems while being run in parallel. For our studies, we consider a multicore machine with 32 cores. This choice of a recent and commonly used system shall enrich the existing literature for multicore systems against the numerous other existing studies over cluster systems. This work is an extension of the preliminary study of the energy-consumption discussed in the previous Section. In this research, we offer a wider comparison between three well-known metaheuristics and their distributed counterparts over a growing number of computing units from 1 to 32 cores. We aim at providing a clear view to the design-functionality of the sequential and distributed algorithms using a different number of cores.

We run our experiments considering the two main termination conditions used in literature. The first termination condition is using the number of function evaluations as a termination condition. The second termination condition is arriving at a specific fitness-value. The scientific contribution of this research is the combined studies for the performance of three distributed algorithms, over a various number of computing units using two different termination conditions. The outcome of these different experiments will vary the novel knowledge of the parallel performance of the different algorithms under the study. We consider running the sequential algorithms [1] and the distributed algorithms using the same parameters over a different number of cores. Each distributed algorithm considers running a single version of the sequential algorithms. The distributed algorithms are working simultaneously in a synchronous manner for finding the optimal solution. Knowledge and information sharing occur every migration gap in a unidirectional ring topology [1]. We present a set of eight various instances: three different cases of the multimodal problem generator (P-PEAKS) [63], and five instances of the Vehicle Routing Problem (VRP) [22].

In the first experiment, we consider using a fixed number of function evaluations, which is widely used as a termination condition in evaluating algorithms. We present the numerical results of the different algorithms, along with their energy consumption and execution time over a varied number of cores vary till 32. We present the average of the obtained fitness-values by the algorithms under the study over a different number of cores in Table 3.9.

The results of Table 3.9 show that the numerical effort of the different distributed implementations is similar for P-PEAKS instances, since all the algorithms were able to find the optimal solution under the same termination condition. Thus, there were no statistical differences that

[1] The results of one core in this work refer to the sequential versions of the algorithms.

UNIVERSIDAD
DE MÁLAGA

Table 3.9: Mean of the obtained fitness values by the algorithms under the study

Problem	Algorithm	# of cores utilized 1	2	4	8	16	32
P-PEAKS20	VNS	0.00	0.00	0.00	0.00	0.00	0.00
	SA	0.00	0.00	0.00	0.00	0.00	0.00
	ssGA	0.00	0.00	0.00	0.00	0.00	0.00
P-PEAKS50	VNS	0.00	0.00	0.00	0.00	0.00	0.00
	SA	0.00	0.00	0.00	0.00	0.00	0.00
	ssGA	0.00	0.00	0.00	0.00	0.00	0.00
P-PEAKS100	VNS	0.00	0.00	0.00	0.00	0.00	0.00
	SA	0.00	0.00	0.00	0.00	0.00	0.00
	ssGA	0.00	0.00	0.00	0.00	0.00	0.00
VRP1	VNS	650.46	684.95	622.14	629.81	604.11	**574.20**
	SA	618.94	**523.07**	558.58	554.33	561.50	551.04
	ssGA	538.55	601.23	555.43	559.74	534.15	**515.03**
VRP2	VNS	932.14	928.27	935.65	902.11	862.19	**843.95**
	SA	986.74	822.05	**784.70**	818.17	790.90	787.64
	ssGA	967.02	905.16	899.57	**877.40**	**884.49**	893.70
VRP3	VNS	**1,015.74**	1,020.33	1,095.45	1,110.77	1,054.09	1,036.19
	SA	919.81	**916.24**	899.72	952.87	978.24	983.17
	ssGA	1,078.34	1,056.16	1,052.88	**1,035.27**	1,166.64	1,207.54
VRP4	VNS	1,437.81	1,566.85	1,489.57	1,493.15	**1,421.49**	1,486.33
	SA	1,218.26	**1,204.54**	1,237.03	1,361.52	1,562.99	1,630.05
	ssGA	1,598.48	**1,559.72**	**1,658.37**	1,762.39	1,965.58	2,167.91
VRP5	VNS	**1,761.29**	1,796.95	1,839.76	1,825.34	1,868.10	1,917.64
	SA	1,567.87	**1,507.02**	1,612.25	1,728.23	2,115.43	2,241.39
	ssGA	2,294.37	**2,147.54**	2,366.41	2,514.85	2,775.86	3,197.84

Boldfaced and underlined represent the least and highest values

could show between the different implementations. The canonical search behavior of the algorithms under the study (without local search) could be effective in solving several instances of benchmark problems. Using more cores improve the numerical results (shown in bold) in most of the cases of VRP instances under the study. However, some cases of the 32 cores algorithms exhibit lower fitness values. Statistically, all the distributed algorithms exhibit a different numerical performance behavior from the sequential ones in most of VRP instances (except two cases of ssGA in VRP2 and VRP4) under the study. That search behaviour is prospective, since all the algorithms under the study consider the same total number of function evaluations as a termination condition. The numerical performance of distributed algorithms persists highly correlated with the number of cores utilized. Knowledge sharing (migration of solutions) used by the parallel algorithm positively affects the overall numerical results of these algorithms.

The second experiment differs from the former one in the stopping condition. Since all the algorithms should arrive at the same specific fitness value, we discuss the numerical performance from the number of function evaluations applied point of view. Table 3.10 presents the average number of function evaluations over 30 independent runs of the algorithms for finding a predefined fitness value.

Since all the algorithms under study were able to arrive at the target fitness value, therefore the resulting numerical effort depends on the design functionality of each algorithm. Despite ssGA goes over the search space with a population of solutions, ssGA and its distributed versions were able to reach the optimal-predefined values with a less numerical effort than the other single-solution based algorithms in many cases. This search behavior proves the efficiency of exploring the search space with a population of solution, rather than using a single solution. The fact that the

UNIVERSIDAD DE MÁLAGA

Table 3.10: Mean number of function evaluations performed by the algorithms

Problem	Algorithm	# of cores utilized 1	2	4	8	16	32
P-PEAKS20	VNS	**12,974.36**	19,374.62	44,594.20	86,668.80	148,491.20	319,708.24
	SA	**6,980.36**	16,196.58	28,154.30	56,627.50	112,963.70	198,573.56
	ssGA	**2,824.65**	5,067.96	8,477.20	16,616.30	32,626.50	59,163.21
P-PEAKS50	VNS	**19,307.65**	48,125.20	51,656.80	92,610.80	234,487.80	316,384.54
	SA	**7,901.38**	14,048.80	28,385.60	55,826.20	111,253.00	196,254.66
	ssGA	**2,468.32**	**4,627.60**	8,364.60	15,760.60	32,886.20	51,836.71
P-PEAKS100	VNS	**11,084.94**	15,257.00	62,295.60	89,006.80	114,087.00	289,173.69
	SA	**7,491.06**	13,843.00	28,219.60	56,658.60	111,837.20	201,624.66
	ssGA	**3,126.57**	4,686.40	8,706.00	16,246.20	32,402.60	60,399.27
VRP1	VNS	**12,961.54**	47,796.70	69,153.00	93,192.10	209,307.40	291,436.21
	SA	**9,834.05**	17,550.60	32,620.20	65,461.70	136,782.90	187,346.32
	ssGA	**41,697.25**	**36,592.10**	47,270.20	82,410.80	131,256.30	215,674.32
VRP2	VNS	**21,687.32**	48,859.50	89,026.20	194,589.00	153,436.30	298,364.10
	SA	**16,078.34**	31,190.20	66,597.30	14,5352.60	241,636.50	376,304.67
	ssGA	**84,361.05**	99,283.70	180,070.30	280,279.30	454,657.20	651,139.25
VRP3	VNS	19,076.20	**2,545.10**	5,298.10	10,324.40	20,120.50	38,719.60
	SA	**11,687.32**	21,263.00	44,375.50	86,440.60	175,338.00	304,367.26
	ssGA	**61,497.36**	**59,708.70**	80,262.10	149,467.50	246,866.50	462,843.02
VRP4	VNS	3,604.95	**2,506.50**	4,701.00	8,486.20	17,406.80	32,618.56
	SA	12,604.61	**11,143.40**	22,294.40	43,456.70	88,197.70	154,039.57
	ssGA	**4,138.78**	**5,064.40**	7,552.50	14,165.60	26,834.00	49,251.65
VRP5	VNS	23,264.98	**13,299.50**	24,138.80	38,407.80	76,853.00	183,673.06
	SA	36,271.08	**28,161.20**	54,879.00	113,256.50	237,492.20	431,094.71
	ssGA	82,0371.69	**53,439.10**	104,163.40	165,612.00	291,093.00	574,106.36

Boldfaced and underlined represent the least and highest values

sequential algorithms scored a lower number of function evaluations (in bold) exist in the serial execution of the sequential algorithms (one solution/population per generation), against a set of distributed solutions/populations every generation simultaneously (the same evaluations phase in each computing unit) in the distributed versions.

The statistical analysis of the results of the number of function evaluations shows that the distributed algorithms exhibit different numerical performance, except four cases of ssGA implementations. This fact confirms the different search behavior of the resulting distributed algorithms. The number of function evaluations is vital for comparing the numerical performance of the algorithms; however, it may be deceiving if used to compare the numerical performance of the sequential/distributed or single-solution/population-based metaheuristics. In this work, we also presented an extensive study to the energy consummation and execution times of the algorithms under the study. For the full results and discussion, please refer to the publication in the corresponding publishing-house.

UNIVERSIDAD DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD
DE MÁLAGA

Chapter 4

Conclusions and Future Works

This thesis presents a coherent study on the numerical performance and energy consumption of different metaheuristics. Metaheuristics are powerful techniques for solving optimization and search problems. Such techniques aim at providing efficient solutions for current era problems within feasible time limits. We present numerous computational experiments considering the scalability in problem size. We measured and analyzed the energy and time consumption behavior of sequential and parallel metaheuristics, two important paradigms of optimization, search, and learning algorithms. Over our experiments, we have used various problems (vary in characteristics of the search space and fitness function complexity) over a varied number of cores and dimensions to expose the behavior of the algorithms. These contributions fully satisfy the answers of the objective goals on the performance of the parallel algorithms over multiprocessors. Our various experiments and comparisons show the numerical performance, energy consumption, and execution time of the sequential and parallel optimization algorithms. The ultimate outcome of this study will help improving algorithms performance, reduce energy consumption, and gas emissions, which is a serious challenge that requires higher attention from society.

We investigated the effect of synchronism on PGAs over multiprocessors systems, covering the three most common parallel models: Sync, Async, and Master/slave. Our focus was on numerical efficiency and time consumption, to provide a common study on the behavior of parallel GAs over homogeneous shared-memory computing systems. Our hypothesis made for the master-slave model has been confirmed by the results and discussion on it: the master-slave model was able to scale-up well over multiprocessors for a different number of problems. The async implementation outperformed the sync implementation in terms of the execution time. Confirmation came after a detailed statistical analysis: it proved that the async algorithm has a positively different numerical behavior than the sync algorithm when being executed over a multiprocessor system. The results of our experiments proved that the sync/async dGAs have different speed-up behavior when running over multiprocessors system.

Our experiments show that speed-up could be high and even super-linear in multiprocessors for the different dGA algorithms. The master-slave model showed competitive numerical effort with perspective with those of the island model dGA. The master-slave model proved to utilize efficiently new processing units resources that become available nowadays, whereas the effective design of the island-model can efficiently exploit the search space. We claim that the island-based GA is numerically efficient more than the master-slave model over multiprocessors, by assuring the

UNIVERSIDAD
DE MÁLAGA

integration of the decentralized design and migration operator. In summary, such differences between the different parallel implementations over new platforms form novel knowledge resources for the researchers. These outcomes fully cover the effect of synchronism on dGAs search behavior.

Furthermore, we performed many experiments to measure and analyze the energy and time consumption behavior of GA and dGA using different problems over a growing number of cores and dimensions to expose the potential behavior of the algorithms. We observed that the energy consumption of problems varies according to many factors, such as the size of the problem, fitness operator complexity, and parameters used. For the sequential GA, the fitness and genetic operators consume most of the energy and time, while the rest of the algorithm operations (housekeeping) do not take a significant amount of energy in most of the scenarios. The fitness operator of GAs controls the energy and time consumption behavior of GA. Mutation scored higher energy consumption than crossover in all of the problems of the study. Moreover, the analysis of the relation between problem size and energy consumption reveals that the energy consumption percentage consumed by GA operators is varying with the change of the dimension. These percentages will not ever be the same on any laboratory experiment, but they are machine and problem settings dependent. With respect to dGA, the results clearly point to a higher efficiency of the asynchronous version (time and energy). The statistical analysis did also confirm their different energy consumption profiles. We want to remark that the optimal energy consumption in the dGA configuration happened when using a number of islands equal to the number of cores.

For analyzing the performance of the different metaheuristics, we proposed a comparative analysis of the combinatorics of the parallel/distributed execution of optimization algorithms over a multi-core system. For such fair comparison, we deploy two different experiments under the same requirements (hardware, operating system, and programming language). We consider investigating the behavior of the distributed algorithms using two different termination criterion. The first experiment presented here consider using a fixed number of evaluations as a termination condition. The second experiment measures the performance of the algorithms while deploying finding a solution with a predefined quality. The results and discussions presented show the relation between using more computing units on numerical performance, energy-consumption, and execution time behavior. Such results represent an extensive study on the performance of three important state-of-the-art metaheuristic algorithms. The several experiments performed there were designed to investigate the parallel combinatorics with the different metaheuristics considering different optimization problems, including real-life problems. Our results confirm the interconnection between solution quality, energy consumption, and execution time for the considered algorithms. Under different termination criterion, the numerical efficiency of the distributed algorithms was reasonable and competitive compared to the sequential algorithms. While considering a fixed number of function evaluations as a termination condition (the first experiment), the distributed algorithms showed better fitness values than the sequential algorithms. In the same consequence, using more cores steadily decreased energy consumption and execution times. Using multiple cores working concurrently highly reduced the execution time, thus reduce the total energy consumption. The outcome of that experiment proves the efficiency of deploying more cores on the search process and parallel performance in general.

UNIVERSIDAD
DE MÁLAGA

From another experimental design aspect, when using finding a solution with a predefined quality as a termination condition (the second experiment), the numerical performance of the distributed algorithms was reliable with a fine-tuned ration compared to the sequential algorithms, respectively. Using more cores (two, four, and eight cores) leads to better energy consumption and execution times in most of the cases. However, a higher number of cores (16, and 32 cores in our case) caused higher energy consumption and execution times due to the trade-off between energy and time. The combined outcome of these two experiments shows the numerical performance, energy consumption, and execution time combinatorics with the parallel execution of the optimization algorithms. The parallel (i.e., distribution) execution of the algorithms (many evaluation phases concurrently) and knowledge sharing between the distributed computing nodes proved to be a promising approach in reducing the energy consumption of the search techniques. The overall outcome of the two experiments shows that there is a trade-off between energy consumption and execution time from a side with numerical effort and the number of function evaluations from another side. We recommend considering time and energy consumption side-by-side with the number of function evaluations when benchmark or compare the different algorithms. Using the number of function evaluations only would be misleading if used to compare the numerical performance of the different sequential and distributed algorithms.

Our algorithms were coded in C++, and the communications in the distributed algorithms employ MPI interface. C++ is a powerful object-oriented programming language that promotes a universal compiling and execution in almost all the operating systems. The MPI interface promotes the efficient exchange for the information among the distributed CPUs, which makes our implementations fault-tolerant with a higher availability over the different operating systems. This combination of the popular programming language and standard library allows our codes to be extendable and available for a wide range of researches and systems for the future.

For future works, we will investigate the parallel efficiency of metaheuristics over high dimensional problems, considering parameter tuning, and employing different communication schemes. A shift for analyzing parallel efficiency on a massively-parallel computer is highly prospective. We will analyze the energy consumption of such techniques to solve problems in the domain of smart cities. We plan to provide a general framework for designing efficient and energy-aware metaheuristics. Multiprocessors are becoming very handy at this moment, and we need more studies on the behavior of metaheuristics on such platforms. Based on our approach and conclusions, we prospective more attention for the analysis of distributed algorithms in the scientific field in the future.

UNIVERSIDAD DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD
DE MÁLAGA

Appendix A

Resumen en Español

La necesidad de resolver problemas complejos y de alta dimensión es un hecho que motiva constantemente a los investigadores a mejorar y proponer nuevos algoritmos de búsqueda. Tradicionalmente, se utilizan dos enfoques principales para abordar este tipo de problemas: métodos exactos y metaheurísticas [8]. Los métodos exactos permiten encontrar soluciones óptimas, pero a menudo no son aplicables, ya que requieren mucho tiempo para determinados problemas del mundo real (alta dimensionalidad, multimodales...). Por otro lado, las metaheurísticas proporcionan soluciones subóptimas, incluso óptimas en muchos escenarios, en un tiempo razonable [8, 111]. El campo de las metaheurísticas proporciona un amplio conjunto de algoritmos eficientes para obtener soluciones a problemas de optimización complejos. Este campo incluye tanto algoritmos basados en trayectoria, que optimizan una única solución candidata, como técnicas basadas en población, que van explorando el espacio de búsqueda del problema utilizando múltiples soluciones candidatas [111]. Pese a las mejoras aportadas por este tipo de métodos, en problemas de muy alta complejidad y dimensionalidad la mayoría de los algoritmos de búsqueda y optimización tienen un alto coste computacional y necesitan de un tiempo muy alto para conseguir soluciones aceptable para ser utilizadas en el dominio de aplicación. En este contexto, el paralelismo surgió como un mecanismo muy popular para mejorar los algoritmos metaheurísticos [8]. El campo de aplicación de estas técnicas paralelas varía desde optimización combinatoria en bioinformática y energía, hasta economía, ingeniería del software, etc., y, en general, cualquier dominio que necesite soluciones rápidas de alta calidad [111].

El paralelismo se presenta como una manera natural de reducir el tiempo de cómputo, pero también de mejorar la calidad de las soluciones encontradas debido al cambio producido en la dinámica de la búsqueda del algoritmo [1, 8]. La computación paralela y distribuida es actualmente un área de intensa actividad en investigación, motivada por una gran variedad de factores. Siempre ha habido una necesidad importante de encontrar soluciones a problemas computacionales muy complejos. Sin embargo, los recientes avances tecnológicos han aumentado la posibilidad de utilizar computadoras paralelas muy potentes para resolver problemas que no se podían abordar en el pasado [106]. De hecho, la mayor parte de los equipos modernos, como computadoras portátiles, móviles o estaciones de trabajo, están equipadas con unidades de procesamiento con múltiples núcleos. Estas plataformas presentan una arquitectura muy atractiva para diseñar e implementar algoritmos distribuidos eficientes. En primer lugar, el método distribuido resultante tiene un comportamiento diferente del algoritmo secuencial, proporcionando así un esquema de búsqueda diferente. En segundo lugar, hace uso de múltiples unidades de procesamiento

UNIVERSIDAD DE MÁLAGA

simultáneamente para resolver el problema. Esto permite amortiguar los costes computacionales de las metaheurísticas, permitiendo obtener soluciones de alta calidad en menor tiempo [9]. El tercer aspecto se relaciona con el uso de metaheurísticas paralelas en plataformas de cómputo de alto rendimiento, que permiten resolver de manera más rápida y precisa problemas de gran escala que involucran grandes cantidades datos.

Recientemente, la eficiencia energética ha ido ganando interés de los investigadores para optimizar los recursos informáticos, especialmente en centros de datos y en la computación en la nube [2, 42, 58, 71]. Estos centros de datos consumen enormes cantidades de energía eléctrica. Al optimizar el uso de estos recursos, además de ahorrar una gran cantidad en gastos debido al consumo de los equipos informáticos, se protege el medio ambiente de las emisiones contaminantes que provocaría la generación de dicha energía eléctrica. Incluso si la electricidad es una forma de energía ecológica, su generación emite toneladas de dióxido de carbono (CO_2) y dióxido de azufre (SO_2) al medio ambiente, lo que aumenta el calentamiento global. Estos contaminantes representan una gran amenaza para los humanos, los animales y la salud de todos los seres de la Tierra [71]. La paralelización (en concreto el modelo distribuido) es un enfoque prometedor para reducir el tiempo de cómputo y por lo tanto reducir los abrumadores valores de consumo de energía. A pesar de los avances recientes en la ejecución de metaheurísticas en paralelo, la comunidad todavía carece de estudios que analicen y comparen, desde un punto de vista energético, las técnicas de optimización canónicas al ejecutarlas en paralelo. Este tipo de estudio proporcionaría a los investigadores elementos cuantitativos que serán de utilidad para desarrollar técnicas de búsqueda eficientes en términos de energía y tiempo.

Las metaheurísticas paralelas verdes (GPM, por sus siglas en inglés, *Green Parallel Metaheutistics*) son un nuevo concepto que queremos introducir en esta tesis. Es una idea inspirada en dos hechos: (i) las metaheurísticas paralelas podrían servir como herramientas únicas para resolver problemas de optimización complejos pero teniendo en cuenta aspectos relacionados con el ahorro energético y la sostenibilidad, y (ii) las computadoras multiprocesadores se han popularizado como plataformas eficientes para ejecutar este tipo de técnicas tanto en el ámbito académico como en la industria. Sin embargo, la construcción de metaheurísticas verdes paralelas requiere un alto grado de conocimiento en el diseño de técnicas de optimización eficientes, así como del consumo de energía real de cada uno de los componentes involucrados en las técnicas empleadas. El objetivo principal es analizar las metaheurísticas secuenciales y paralelas considerando su consumo de energía. Esto requiere una investigación profunda del rendimiento de estas metaheurísticas, pero el conocimiento obtenido permitirá diseñar en el futuro algoritmos más eficientes. Presentamos un análisis de la mejora obtenida por diferentes metaheurísticas paralelas cuando se ejecutan en plataformas con múltiples procesadores. Además, analizamos el consumo energético de diferentes algoritmos secuenciales/paralelos, así como de sus componentes.

A.1 Organización

Esta tesis está escrita en forma de un compendio de publicaciones. La organización de este resumen es la siguiente: la Sección A.2 presenta el estado del arte en el área de las metaheurísticas. La Sección A.3 presenta los conceptos de las metaheurística paralelas y distribuidas. La Sección A.4 presenta los conceptos de eficiencia energética de los algoritmos. La Sección A.5

UNIVERSIDAD
DE MÁLAGA

presenta un compendio de los trabajos publicados, donde presentamos sus principales contribuciones. Concluimos este documento con las principales aportaciones de la tesis y sus posibles líneas de trabajos futuras.

A.2 Metaheurísticas

Los problemas de optimización son muy habituales en la vida real tanto en ambientes industriales, científicos, como ingenieriles por mencionar algunos posibles dominios de aplicación [47, 69]. Estos problemas son complejos y difíciles de resolver mediante métodos de búsqueda exactos [9]. La necesidad de resolver estos problemas fue el motor de la aparición de una nueva clase de algoritmos llamada “Metaheurísticas” [45]. Las metaheurísticas demostraron ser capaces de resolver grandes problemas en un tiempo razonable y mucho más rápido que las técnicas exhaustivas [111]. En las últimas cinco décadas hemos sido testigos del surgimiento de muchos algoritmos dentro del campo de las metaheurísticas, muchos de estos algoritmos se basan en imitar un fenómeno en la naturaleza o la industria como por ejemplo la evolución genética en la naturaleza o el proceso de enfriamiento en la metalurgia. Las técnicas metaheurísticas incluyen una amplia variedad de diseños y comportamientos durante la búsqueda, lo que ha permitido su aplicación para la resolución de problemas en diferentes campos de investigación. Las metaheurísticas se clasifican principalmente en dos categorías: técnicas basadas en *trayectoria* y técnicas basadas en *población* [8].

Actualmente las metaheurísticas son métodos muy populares para resolver una amplia gama de problemas tanto académicos como del mundo real, pero es realmente difícil encontrar estudios sobre su comportamiento en relación a su consumo energético. La mayoría de las técnicas de búsqueda fueron diseñadas para encontrar una solución adecuada con un costo de cómputo reducido. De acuerdo con este objetivo, el rendimiento numérico de los algoritmos ha crecido constantemente. Sin embargo, la realización de estudios sobre el gasto energético de este tipo de algoritmos ha sido ignorado por la mayoría de los investigadores en el dominio. La medición del consumo energético de cualquier tipo de software aún se enfrenta a muchos problemas prácticos y teóricos, especialmente por la falta de software especializado que sea capaz de medir con precisión el consumo de energía (independiente del hardware) [58]. Incluso, aún disponiendo de dicha información, el diseño y la construcción de software real que la utilice es un reto bastante complejo [116]. Entre los tipos de programas que más consumo podemos encontrar, las técnicas de búsqueda para resolver problemas complejos ocupan un lugar destacado. Estos métodos requieren grandes tiempos de cálculo y se ejecutan con frecuencia para optimizar las actividades diarias en ciudades y fábricas [42]. Todos los problemas de rendimiento y diseño mencionados anteriormente sobre las metaheurísticas han motivado nuestro trabajo en esta tesis.

Por lo tanto en esta tesis, nos enfocamos en realizar estudios sobre la calidad de la solución, el consumo de energía y el tiempo de ejecución de un conjunto significativo de metaheurísticas diferentes y sus versiones distribuidas mientras se ejecuta en entornos informáticos modernos. Nuestros algoritmos incluyen dos enfoques basados en la trayectoria (*Variable Neighborhood Search*, VNS, y, *Simulated Annealing*, SA), un enfoque basado en la población (*Genetic Algorithms*, GA) y sus versiones distribuidas. En cuanto a la plataforma de ejecución nos enfocamos en los equipos multiprocesadores, que están cobrando gran importancia en la actualidad y de los cuales aún existen muy pocos estudios.

UNIVERSIDAD DE MÁLAGA

A.3 Conceptos de metaheurística paralela

Las metaheurísticas (como la mayoría de los métodos de búsqueda y optimización para resolver problemas reales) tiene un alto coste computacional tanto en recursos como en tiempo. En un plazo fijo, debido a su naturaleza estocástica, las metaheurísticas no garantizan llegar a una solución óptima para la mayoría de los problemas de optimización [61]. Se han propuesto muchos enfoques y avances para superar estos inconvenientes. Un enfoque popular es ejecutar los algoritmos en paralelo (por ejemplo, usando el modelo distribuido [8]). Los nuevos algoritmos distribuidos resultantes reducen el tiempo de búsqueda y exhiben nuevas características de búsqueda que difieren de las secuenciales haciéndolos muy atractivos para resolver problemas reales [3, 15].

En la literatura, hay muchas estrategias paralelas para ejecutar algoritmos metaheurísticos (tanto basados en población como basados en trayectoria) en paralelo. Según la fuente del paralelismo, se suelen distinguir tres estrategias de metaheurísticas paralelas [33]:

- **Paralelismo de bajo nivel**: este enfoque consiste ejecutar los cálculos (función de evaluación, búsquedas locales, operadores específicos...) en paralelo con el objetivo de acelerar el proceso de búsqueda. Este tipo de paralelismo intenta reducir el tiempo de ejecución, pero sin cambiar el comportamiento de búsqueda respecto a la versión secuencial. Un modelo típico para este enfoque paralelo es el modelo clásico maestro-esclavo [26].

- **Descomposición de variables de datos y decisión**: en este enfoque paralelo, se considera una descomposición del espacio de búsqueda (dominio) dividiendo el espacio de búsqueda en regiones más pequeñas, aplicando así la metaheurística secuencial en cada región que considera las variables fuera de su subconjunto como fijas. La descomposición del dominio se ha empleado con éxito en el diseño y la ejecución de metaheurísticas en paralelo [115].

- **Múltiples exploraciones concurrentes**: aquí, el paralelismo se obtiene de múltiples recorridos en el espacio de la solución al mismo tiempo. Cada hilo concurrente puede ejecutar exactamente el mismo método heurístico, o uno totalmente diferente. Estos hilos distribuidos se comunican durante la búsqueda intercambiando conocimiento. Un ejemplo de este enfoque paralelo es la ejecución distribuida [1].

En esta tesis nos enfocamos en este último modelo, debido a diferentes motivos: (i) es uno muy habitual en la literatura y por lo tanto nuestras contribuciones pueden ser útiles para un conjunto muy importante de investigadores; (ii) no solo permite mejorar la eficiencia del algoritmo sino también su eficacia debido a los cambios que se producen en el espacio de búsqueda al explorar en paralelo diferentes regiones y compartir dicha información; y (iii) son muy adecuado para plataformas hardware muy diferentes , incluyendo los equipos multiprocesadores, que es otro de los elementos diferenciadores de esta tesis.

A.3.1 Esquemas de comunicación

Las metaheurísticas se pueden paralelizar dividiendo las tareas del algoritmo distribuidas en diferentes procesadores. Estos procesadores diferentes se comunican para compartir conocimientos y datos obtenidos durante la búsqueda. Un tipo común de GA paralelos (PGA) es el *modelo*

UNIVERSIDAD DE MÁLAGA

de isla, también conocido como *modelo distribuido* [8]. En este modelo, la población global se divide en subpoblaciones (islas) distribuidas en diferentes procesadores. Estas islas ejecutan un GA básico cada una, y después de un intervalo predefinido se comunican para intercambiar el conocimiento de búsqueda, principalmente individuos o parámetros. La comunicación entre las islas se denomina “migración”. El operador de migración se encarga del intercambio de individuos entre las subpoblaciones del dGA (GA distribuido). La topología de migración considerada en la tesis se basa en la topología de anillo unidireccional analizada en muchos trabajos [17, 103, 107]. En esta topología, las comunicaciones entre las islas se realizan en un anillo unidireccional, como se muestra en la Figura A.1, con el fin de mantener un bajo coste relacionado con las comunicaciones general de la dGA y para permitir un difusión más suave de la información entre las islas.

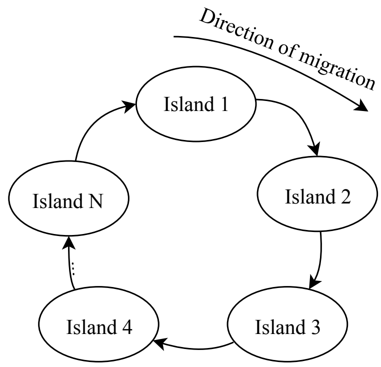


Figure A.1: Topología de anillo unidireccional para la migración entre subpoblaciones

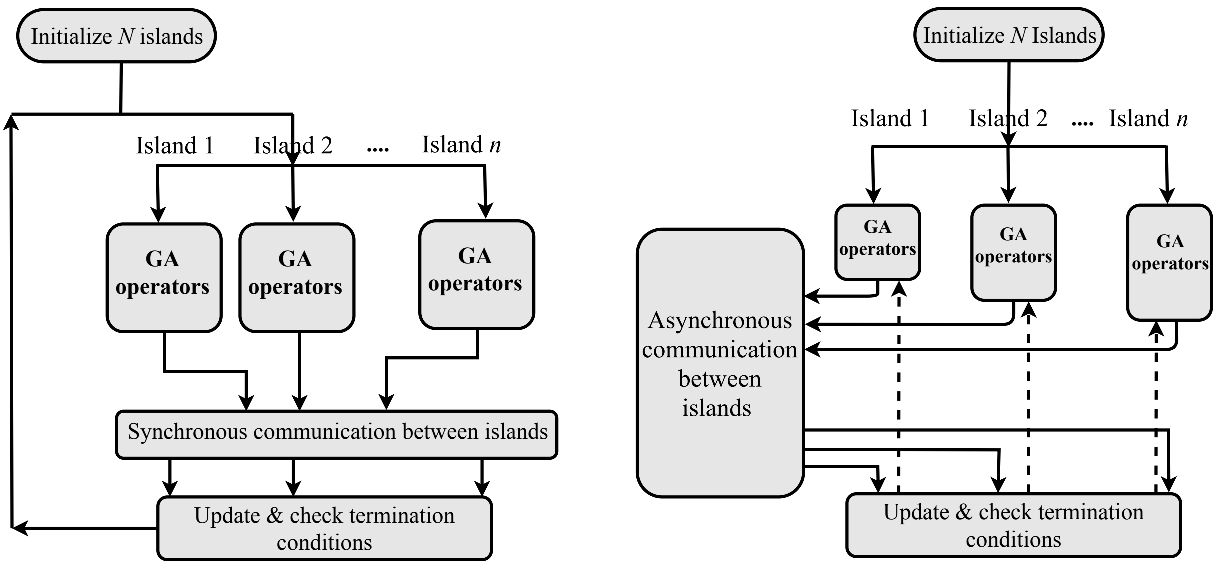
En nuestros estudios, cada subpoblación envía y recibe el mejor individuo de la subpoblación vecina, y reemplaza al peor de su grupo. Expresamos el intervalo de migración en términos del número de evaluaciones realizadas en una subpoblación. Se eligió el intervalo de migración para permitir que cada proceso realice una exploración suficiente por sí solo entre cada par de intercambio de información. Además, la tasa de migración se determinó como un número fijo de los mejores individuos de la población en cada migración, para permitir un estudio más estructurado.

A.3.2 Implementaciones síncronas y asíncronas

La ejecución paralela con esquemas de comunicación tal como el propuesto en el apartado previo proporcionan a la metaheurísticas unas características de búsqueda potencialmente diferentes. En la literatura, podemos encontrar dos esquemas de comunicación muy populares, la comunicación síncrona y asíncrona. En el enfoque síncrono se tiene un punto de sincronización en cada intervalo de migración, donde ocurren las comunicaciones [6]. En este punto, todos los procesos deben esperar y bloquearse hasta que todos los procesos alcancen el mismo punto de su código. Por lo tanto, es muy probable que en cada punto de comunicación haya algunos procesos inactivos esperando que llegue el resto para poder realizar la migración. Por el contrario, en el enfoque asíncrono, no existen tales puntos de sincronización, ya que las subpoblaciones incluyen a los individuos recibidos siempre que sea posible, evitando así cualquier bloqueo [14] y por lo tanto, no hay procesos inactivos. El diseño del dGA asíncrono está desarrollado para permitir interacciones eficientes y promover la superposición de cálculos y comunicaciones [74]. En nuestros estudios, las

UNIVERSIDAD DE MÁLAGA

implementaciones, que denominamos **Sync** y **Async**, ejecutan el mismo algoritmo con la misma configuración siendo la única diferencia en el esquema de comunicación. La Figura A.2 muestra las diferencias entre los dos esquemas.



(a) Esquema síncrono

(b) Esquema asíncrono

Figure A.2: Diagrama funcional para un dGA con los esquemas síncrono y asíncrono

A.4 Eficiencia energética de los algoritmos

La variedad de técnicas equipadas con múltiples operadores de búsqueda le da a las meta-heurísticas un amplio campo de aplicabilidad para resolver problemas de optimización y búsqueda [111]. La técnica de búsqueda básica de estos algoritmos fue diseñada para ejecutarse secuencialmente con el objetivo principal de encontrar una solución adecuada y reducir el coste de cálculo. De acuerdo con este objetivo, el rendimiento numérico de los algoritmos ha ido creciendo constantemente. Sin embargo, la eficiencia y la relación entre la ejecución paralela y el consumo de energía del algoritmo se ha dejado bastante desatendida.

Hoy en día, el enorme consumo de electricidad y la emisión de dióxido de carbono están motivando a los científicos y a la sociedad a buscar aplicaciones más ecológicas (que consuman menos energía). El objetivo es avanzar hacia un nuevo diseño de algoritmos que agregue eficiencia energética en sinergia con el rendimiento numérico. Esta interconexión del consumo de energía y la eficiencia numérica del algoritmo motiva un mayor interés en el estudio y el análisis de los algoritmos de búsqueda de última generación al considerar su consumo de energía. Se pueden obtener dos resultados principales al mejorar la eficiencia energética de los algoritmos de búsqueda. El primero es que los algoritmos que consideren la eficiencia energética reducirán el impacto ambiental. El segundo se relaciona con un proceso de búsqueda más rápido que también mejore el confort en su entorno (las máquinas producen menos sobrecarga de calor, por lo tanto, menos costes y ruido del sistema de enfriamiento) [87].

UNIVERSIDAD DE MÁLAGA

Para nuestros estudios, consideramos una máquina multinúcleo con 32 núcleos. Esta elección de un sistema moderno y de uso común enriquecerá la literatura existente para sistemas multinúcleo, que está bastante descuidado frente a la gran cantidad de estudios que existen para sistemas basados en clústeres de ordenadores. En una máquina multinúcleo homogénea, todos los nodos informáticos tienen las mismas características, lo que significa que no hay retraso para esperar nodos más lentos. Además, el uso de un sistema de memoria compartida reducirá los costes de comunicación. En base a estos factores, la ejecución paralela de los algoritmos bajo estos sistemas mostrará diferentes perfiles de rendimiento. En la literatura actual, hay pocas investigaciones que estudian la eficiencia de los algoritmos en multiprocesadores [1]. Nuestro enfoque en esta tesis es recopilar los valores de consumo de energía de las diferentes metaheurísticas en tiempo de ejecución. Para ello usamos un enfoque software para recopilar los valores de consumo de energía. En concreto, consideramos una biblioteca de monitoreo del sistema (a nivel de lenguaje de programación) para hacer uso de contadores de CPU llamada RAPL (*Running Average Power Limit*) [35].

La mayoría de los estudios anteriores que consideran la medición del consumo de energía se basaron en medidores físicos acoplados al enchufe o sensores de medición internos [67, 95]. Los medidores acoplados al enchufe miden el consumo de energía de todo el sistema, no el proceso de ejecución. Los sensores de medición internos varían según el tipo y el método de estimación del consumo de energía. Los valores de medición obtenidos por estos métodos generalmente se combinan con una alta sobrecarga de los otros procesos en ejecución o el calentamiento térmico de la máquina [20, 52, 92]. RAPL estima el consumo de energía de un proceso en ejecución, en tiempo real, en función de la información recopilada de los contadores de registro específicos del modelo (MSR) (por ejemplo, CPU o RAM) a través del sistema operativo. Estudios recientes que investigan la eficiencia de RAPL confirmaron que los valores proporcionados por RAPL son altamente precisos [101] y están altamente correlacionados con la potencia del sistema añadiendo una sobrecarga insignificante en el sistema [64].

A.5 Resumen de resultados

Esta tesis doctoral está avalada por cinco contribuciones de alta calidad, tres de ellas publicadas en revistas indexadas en el JCR y dos en conferencias internacionales de alto prestigio. El resumen de estas publicaciones que se presenta en las siguientes secciones sigue el siguiente orden:

[1] Amr Abdelhafez and Enrique Alba. **Speed-up of synchronous and asynchronous distributed genetic algorithms: A first common approach on multiprocessors.** 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2677-2682, 2017.

Swarm and Evolutionary Computation: Q1, Impact factor: **6.33**.

[2] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors.** Swarm and Evolutionary Computation, 49:147-157, 2019.

The Journal of Supercomputing: Q2, Impact factor: **2.16**.

[3] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **A component-based study of energy consumption for sequential and parallel genetic algorithms.** The Journal of Supercomputing, 75(10):6194-6219, 2019.

UNIVERSIDAD DE MÁLAGA

[4] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Analyzing the energy consumption of sequential and parallel metaheuristics.** 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019.

Swarm and Evolutionary Computation: Q1, Impact factor: **6.33**.
[5] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Parallel Execution Combinatorics with Metaheuristics: Comparative Study.** Swarm and Evolutionary Computation, 2020.
DOI: https://doi.org/10.1016/j.swevo.2020.100692

A.5.1 Speed-up of Synchronous and Asynchronous Distributed Genetic Algorithms: A First Common Approach on Multiprocessors

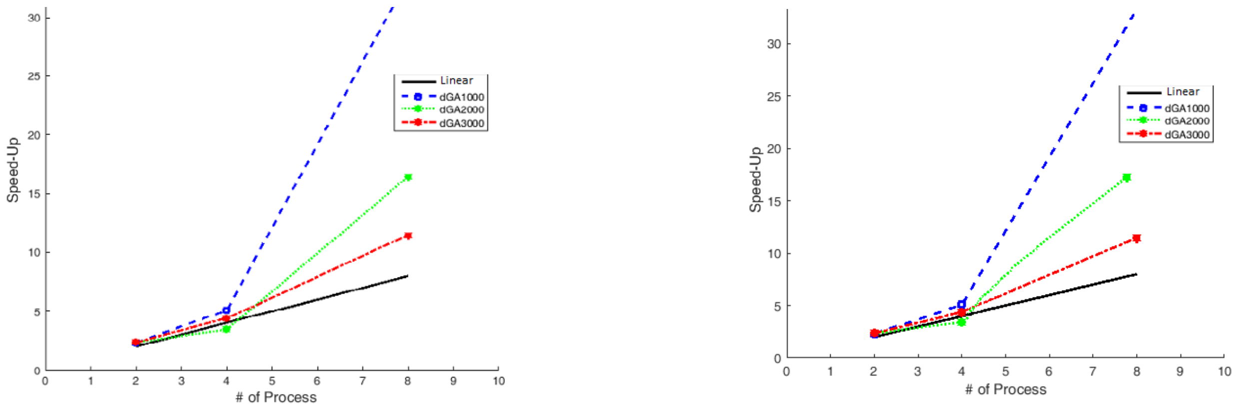
Los algoritmos genéticos se están utilizando para resolver una amplia gama de problemas del mundo real, por lo que es importante estudiar sus implementaciones para mejorar la calidad de la solución y reducir el tiempo de ejecución. El diseño de GA paralelos (por ejemplo siguiendo el modelo distribuido) es una línea de investigación con mucho potencial [111]. En este artículo se presentó un estudio básico sobre la ganancia en velocidad de los GA paralelos en el que se sigue un enfoque unificado para comprender mejor las versiones síncronas y asíncronas. Analizamos el comportamiento de los GA sobre un sistema multiprocesador homogéneo. Se reportó resultados que muestran una ganancia lineal e incluso superlineal en ambos casos de estudio.

La contribución científica del trabajo radica en el estudio combinado de las variantes síncronas y asíncronas de un dGA desde un tiempo de ejecución y un punto de vista numérico sobre multiprocesadores. Al contrario de la mayoría de los otros estudios, ejecutamos los algoritmos en un sistema multiprocesador, que potencialmente puede difuminar las diferencias numéricas y de tiempos de ejecución debido a la homogeneidad de la plataforma. Dado que el objetivo de este estudio es el algoritmo paralelo y su comportamiento numérico en un sistema multiprocesador, utilizamos un problema común y bien conocido como es el ONEMAX para este primer estudio. Nuestra topología de migración se basa en la topología de anillo unidireccional descrita en [107], donde se clasificó como una buena propuesta entre las 14 topologías de migración diferentes estudiadas. En nuestros experimentos, abordamos implementaciones síncronas y asíncronas utilizando los mismos parámetros para el mismo problema. Realizamos nuestros experimentos en un multiprocesador dedicado homogéneo sin interrupciones de ningún otro proceso.

La Figura A.3 muestra la ganancia en tiempo (*speedup*) de nuestros algoritmos para instancias del problema ONEMAX con dimensiones 1000, 2000 y 3000. La figura muestra claramente que nuestras técnicas constantemente logran una buena ganancia en todas las dimensiones estudiadas para incluso en el número menor de núcleos utilizados en este experimento.

Estos resultados demuestran que las dos implementaciones son igualmente efectivas y rápidas desde una perspectiva de esfuerzo numérico, que no siempre es el caso en otras plataformas paralelas como son los clústeres de computadoras [14]. Tanto en casos síncronos como asíncronos, nuestro algoritmo obtuvo una alta eficiencia para encontrar la solución óptima para problemas de alta dimensión. Por lo tanto, el rendimiento paralelo de las versiones síncronas y asíncronas es muy bueno en una computadora multiprocesador, tanto en términos de tiempo como de calidad de la solución. Finalmente descubrimos que la ganancia es muy alta e incluso superlineal.

UNIVERSIDAD DE MÁLAGA



(a) Modelo síncrono (b) Modelo asíncrono

Figure A.3: Ganancia en velocidad de los esquemas síncrono y asíncrono del dGA

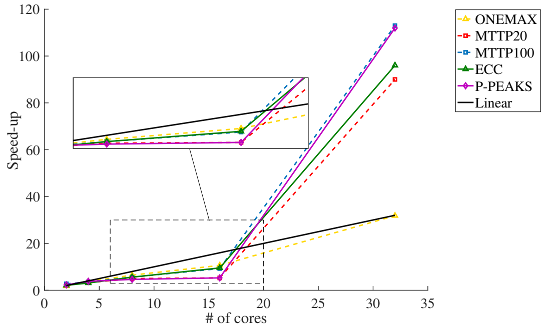
A.5.2 Performance Analysis of Synchronous and Asynchronous Distributed Genetic Algorithms on Multiprocessors

Este artículo es una extensión de la investigación descrita en la sección anterior, donde se obtuvieron resultados preliminares de un dGA aplicado a un único problema (ONEMAX) y con ocho núcleos a lo sumo. En esta investigación, ampliamos nuestro estudio preliminar analizando nuevos problemas (ONEMAX, MTTP, ECC y P-PEAKS). Estos problemas poseen diferentes espacios de búsqueda y diferentes dimensionalidades. Primero, analizamos uno de estos problemas con varios tamaños de dimensión. En segundo lugar, estudiamos el efecto de usar un mayor número de núcleos, variándolos desde uno a 32 núcleos. Con esto pretendemos mostrar el efecto del sincronismo en los algoritmos distribuidos, utilizando un número diferente de problemas y dimensiones en un número variable de núcleos. Además, estos resultados y ganancias se comparan con los obtenidos por otro modelo paralelo clásico como es el esquema maestro-esclavo. La contribución científica de este artículo consiste proporcionar un estudio combinado de diferentes modelos paralelos para GA desde un punto de vista tanto numérico como en tiempo de ejecución. Nuestro objetivo es facilitar a los investigadores y usuarios finales una fuente de información fiable que les permita desarrollar una técnica de búsqueda eficiente que se ajuste a sus propios objetivos. La Figura A.4 ofrece una visualización gráfica de los resultados.

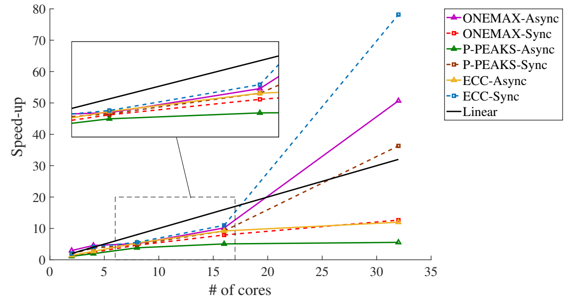
El modelo maestro-esclavo logra constantemente una mayor velocidad para todos los problemas del estudio (Figura A.4 (a)). Las figuras A.4 (b) y A.4 (c) muestran la ganancia de velocidad de las implementaciones síncronas y asíncronas. Los algoritmos síncronos (en líneas punteadas) tienen un tiempo de ejecución diferente y, por lo tanto, un comportamiento de ganancia diferente a los asíncronos (líneas continuas) sobre multiprocesadores.

En resumen, para todas las implementaciones del dGA en el estudio el uso de más núcleos permitió mejorar la ganancia, lo que es bueno para la escalabilidad de estos algoritmos en sistemas de múltiples procesadores. El modelo maestro-esclavo mostró un esfuerzo numérico competitivo respecto a los modelos basados en islas. También este modelo basado maestro-esclavo demostró utilizar eficientemente nuevos recursos de unidades de procesamiento que están disponibles hoy en

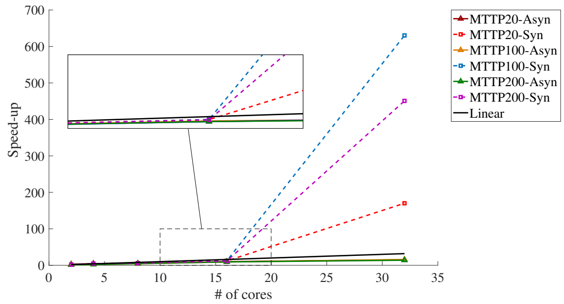
UNIVERSIDAD DE MÁLAGA



(a) Modelo maestro-esclavo



(b) Sync/async dGAs (ONEMAX, P-PEAKS, and ECC)



(c) Sync/async dGAs (MTTP20, MTTP100, and MTTP200)

Figure A.4: Ganancia de diferentes implementaciones del dGA

día, mientras que un diseño efectivo del modelo basado en islas permite explotar eficientemente el espacio de búsqueda. Los resultados confirman que el GA basado en islas es numéricamente más eficiente que el modelo maestro-esclavo cuando se utilizan plataformas con multiprocesadores, al utilizar la integración del diseño descentralizado y el operador de migración.

A.5.3 A Component Based Study of Energy Consumption for Sequential and Parallel Genetic Algorithms

Las principales contribuciones de este trabajo son estudiar el consumo de energía y el comportamiento del tiempo de ejecución de GAs secuenciales y paralelas, en un análisis de energía razonablemente amplio de sus componentes bajo diferentes esquemas de comunicación. El impacto potencial de estos resultados en la construcción de nuevas técnicas que se ajusten a los objetivos de la computación ecológica también se vinculará a una línea de investigación para hacer que los algoritmos sean más eficientes como un software que se ejecuta en una computadora. Este punto de vista no está tan presente en la literatura, pero es fundamental para formular investigaciones de alta calidad.

En un primer experimento, analizamos el consumo de energía de un GA panmíctico secuencial. La Figura A.5 presenta los porcentajes de consumo de energía de los componentes del algoritmo.

La Figura A.5 muestra que el operador de evaluación es el que consume más energía para cuatro problemas, mientras que en los otros cuatro problemas el operador que consume más

UNIVERSIDAD DE MÁLAGA

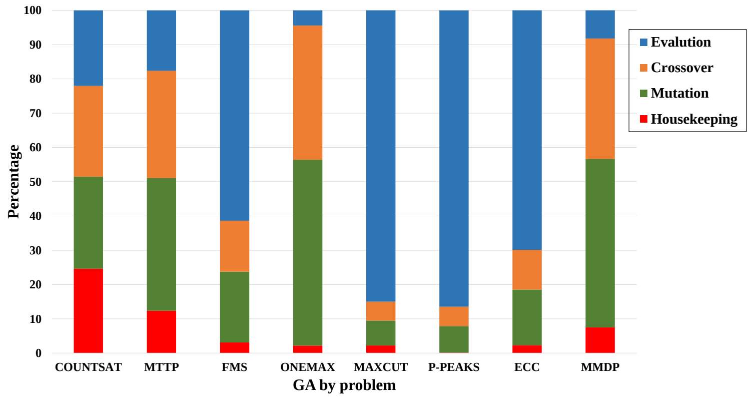


Figure A.5: Porcentajes de consumo de energía (%) de los componentes del GA

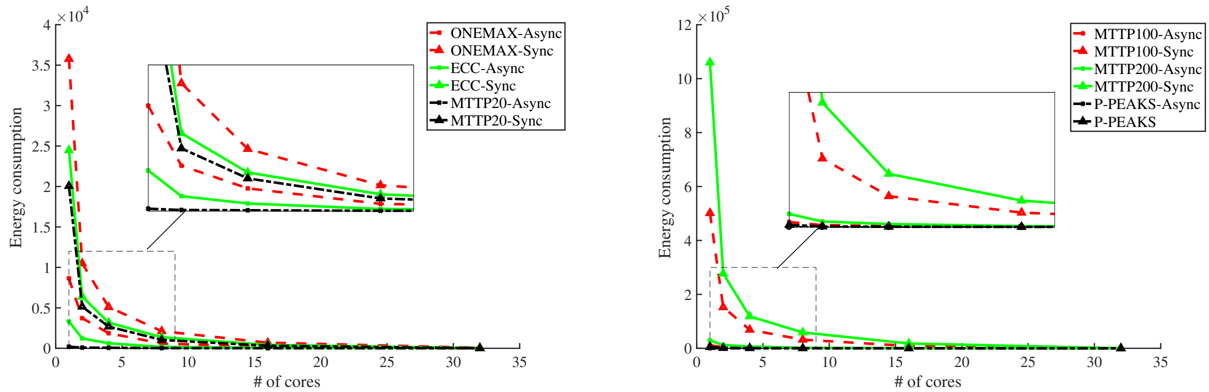
energía es la mutación. También confirma otra observación importante: que la mutación representa un componente potencial costoso desde el punto de vista energético dentro del GA. El consumo de mutaciones es, sin embargo, más sutil. Este operador genera constantemente números aleatorios para decidir si un bit debe modificarse o no. Se demostró que la generación de números aleatorios es un operador con bastante coste (en tiempo y energía) y esa es la razón del comportamiento de consumo de mutaciones. Las operaciones de mantenimiento representan los porcentajes de energía más pequeños en siete de nuestros ocho problemas.

En otro experimento, presentamos un análisis de los modelos paralelos dGA síncronos y asíncronos. Ambas implementaciones tienen los mismos parámetros y la única diferencia es su esquema de comunicación. Nuestro algoritmo consta de 32 islas para todos los experimentos realizados, y estudiamos un número creciente de núcleos para ejecutar estas 32 islas, de uno a 32 núcleos.
La Figura A.6 muestra el comportamiento del consumo energético de los algoritmos síncronos y asíncronos, respectivamente, en un número diferente de núcleos. Presentamos los resultados para seis instancias: ONEMAX de tamaño 2000 bits, P-PEAKS de 100 bits, ECC de tamaño 288 bits y tres instancias de MTTP denominadas MTTP20, MTTP100 y MTTP200.

Los resultados de la Figura A.6 apuntan claramente al hecho de que la implementación asíncrona es más eficiente que la implementación síncrona: consume menos cantidad de energía y tiempo en la mayoría de los casos del estudio (para el mismo rendimiento numérico). En términos de tiempo de ejecución, este comportamiento se informó en muchos estudios anteriores sobre el sincronismo de GA, por ejemplo, [14]. Nuestros resultados confirman también este bajo rendimiento de la implementación síncrona en términos de consumo de energía.

En resumen, nuestros experimentos en los GAs secuenciales muestran el papel controlador del operador de evaluación en el consumo de energía. También revela posibles puntos críticos relativos al consumo energético en las operaciones del GA, como el operador de mutación. La mutación se mostró como el componente de mayor consumo en cuatro de los problemas del estudio. La mutación obtuvo un mayor consumo de energía que el cruce en todos los problemas del estudio. En este trabajo, también presentamos un análisis de la relación entre el tamaño del problema y el consumo de energía. El análisis revela que el porcentaje de consumo de energía consumido por los operadores de GA varía con el cambio de dimensión. Nuestras evaluaciones distribuidas

UNIVERSIDAD
DE MÁLAGA



(a) Problemas: ONEMAX, ECC, and MTTP20 (b) Problemas: MTTP100, MTTP200, and P-PEAKS

Figure A.6: Consumo de energía (en kWh) de los algoritmos síncrono y asíncrono

demuestran que el esquema de comunicación podría afectar en gran medida el consumo de energía de las evaluaciones paralelas del GA. Con respecto a dGA, los resultados apuntan claramente a la mayor eficiencia de la versión asíncrona (tiempo y energía), que notamos para todas las cantidades de núcleos. El análisis estadístico también confirmó sus diferentes perfiles de consumo de energía.

A.5.4 Analyzing the Energy Consumption of Sequential and Parallel Metaheuristics

Este trabajo presenta dos nuevos experimentos para investigar el rendimiento numérico y la eficiencia energética de metaheurísticas secuenciales y paralelas. El objetivo principal de este estudio es analizar el consumo de energía de tres metaheurísticas conocidas y de uso común: GA, VNS y SA, y sus versiones paralelas ejecutadas en 32 núcleos. Por lo tanto, proponemos un análisis del rendimiento de las metaheurísticas secuenciales y distribuidas utilizando las mismas condiciones: la misma condición de parada, que usen el mismo equipo de hardware, enfoque de software y configuraciones de problemas. En la literatura, hay dos condiciones de parada principales. La primera condición es usar el número de evaluaciones como una terminación. La segunda condición de parada consiste en continuar la ejecución hasta alcanzar un valor de aptitud específico (óptimo o alguno que se considere lo suficientemente bueno).

En un primer experimento investigamos la relación entre el consumo de energía, el tiempo de ejecución y la calidad de la solución cuando se utiliza un número fijo de evaluaciones como condición de parada. Posteriormente, realizamos un segundo experimento analizando el rendimiento de los algoritmos cuando llegamos a un valor de aptitud específico como la condición de terminación. Las contribuciones de este trabajo son el análisis del desempeño de seis algoritmos diferentes, atendiendo a sus perfiles de consumo de energía. Estudiamos el rendimiento de los algoritmos considerados mediante el uso de diferentes problemas tanto de académicos como del mundo real. Consideramos un conjunto de ocho instancias: tres casos diferentes del generador de problemas multimodales (P-PEAKS) [63], y cinco instancias del problema de enrutamiento de una flota de vehículos (VRP) [22].

UNIVERSIDAD DE MÁLAGA

La Figura A.7 presenta los valores promedio del consumo de energía (en Julios) de los algoritmos para el primer experimento. Los términos dVNS, dSA y dssGA se refieren a las versiones distribuidas.

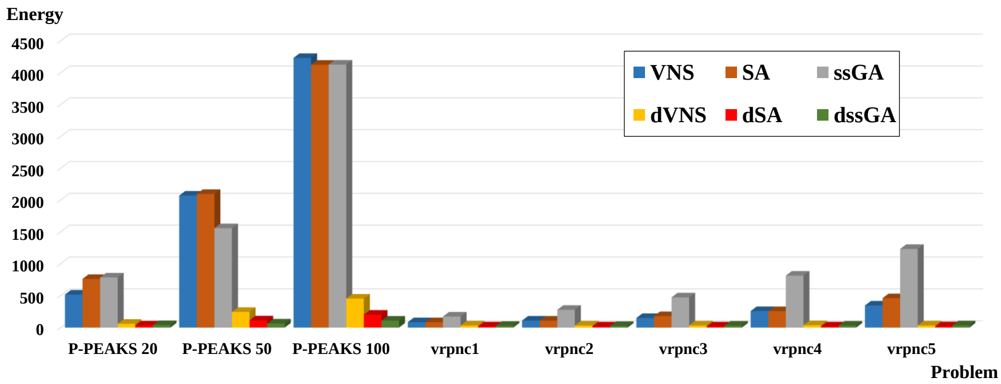


Figure A.7: Consumo de energía en Julios para los algoritmos en estudio (Exp. 1)

El paralelismo permite ejecutar varios pasos algorítmicos al mismo tiempo, reduciendo de manera significativa la ejecución y, en consecuencia, la energía consumida como se muestra en la Figura A.7. El consumo energético debido al uso de más núcleos se compensa en gran medida por la reducción del tiempo de uso de estos recursos. La diferencia en la energía consumida por los algoritmos depende de las características especiales de cada uno, como el operador de cruce en ssGA o el cálculo del criterio de aceptación en SA.

En el segundo experimento, implementamos otra condición de terminación: encontrar una solución con una calidad predefinida para todos los algoritmos, con un máximo de evaluaciones de función 10^6. Determinamos estos valores predefinidos mediante un conjunto de experimentos preliminares, para asegurarnos de que todos los algoritmos llegarían a esta aptitud objetivo. La Figura A.8 presenta el consumo medio de energía de los algoritmos. Los resultados son el promedio de 30 ejecuciones independientes, donde todos los algoritmos pudieron alcanzar la solución óptima o el valor de aptitud predefinido.

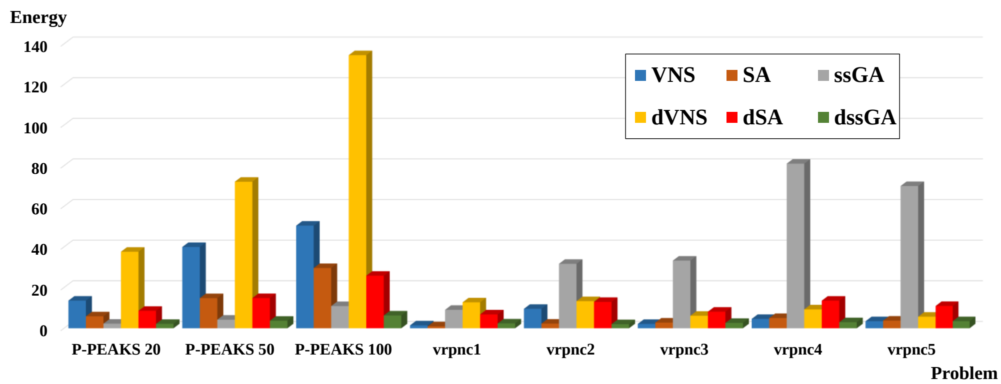


Figure A.8: Consumo de energía en Julios para los algoritmos en estudio (Exp. 2)

La ejecución distribuida y la cooperación entre islas son las claves principales para reducir el esfuerzo numérico, reduciendo así el tiempo de ejecución y el consumo de energía de las

UNIVERSIDAD DE MÁLAGA

técnicas de búsqueda. El ssGA utiliza una población de soluciones, por lo que se espera una exploración más amplia del espacio de búsqueda. La Figura A.8 muestra claramente que dVNS y VNS consumieron los valores de energía más altos para encontrar una solución para el problema P-PEAKS. P-PEAKS posee un espacio de búsqueda multimodal, por lo que el VNS básico (sin búsqueda local) requiere un mayor esfuerzo numérico para encontrar la solución óptima. Para el VRP, ssGA obtuvo los valores más altos de consumo de energía en la mayoría de los casos. El uso de una población secuencial de soluciones en ssGA es la clave principal para este comportamiento de alto consumo de energía.

En resumen, los resultados y las discusiones de los diferentes experimentos muestran y confirman la relación entre el consumo de energía, el tiempo de ejecución y la calidad de la solución para los algoritmos considerados. La ejecución paralela de los algoritmos y el intercambio de conocimientos entre los algoritmos distribuidos demostró ser un enfoque prometedor para reducir el consumo de energía de las técnicas de búsqueda. El resultado general de los dos experimentos muestra una compensación entre el consumo de energía y el tiempo de ejecución de un lado con esfuerzo numérico, y el número de evaluaciones de funciones de otro lado.

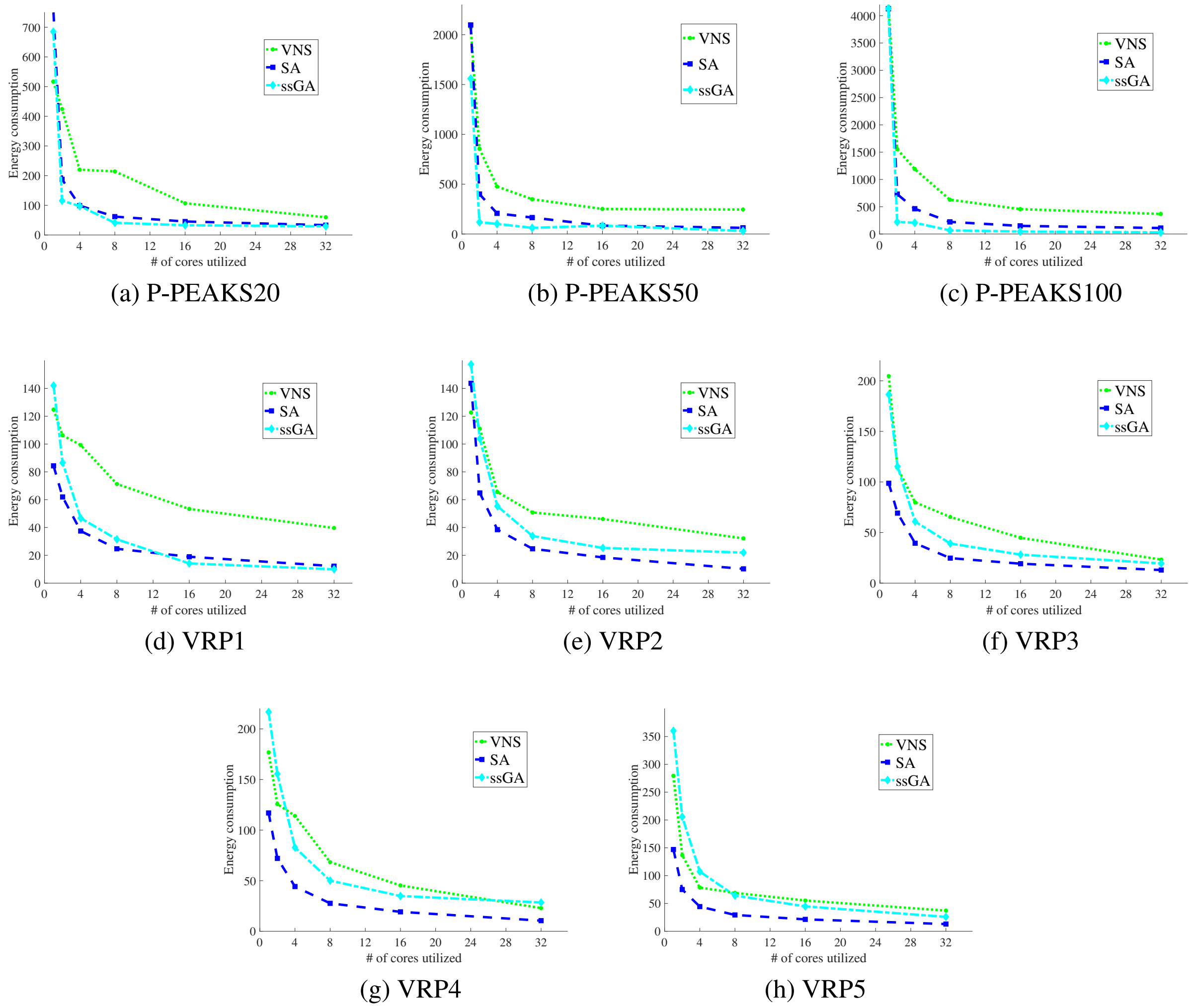
A.5.5 Parallel Execution Combinatorics with Metaheuristics: Comparative Study

En este trabajo presentamos dos amplios estudios sobre la calidad de la solución, el consumo de energía y el tiempo de ejecución de tres metaheurísticas diferentes (GA, VNS y SA) y sus versiones distribuidas. El objetivo principal de nuestro estudio es explorar la eficiencia de la ejecución paralela de la metaheurística mientras se ejecuta en una máquina multinúcleo. Este estudio difiere de los estudios explicados en las secciones anteriores en los criterios de terminación y la comparación entre diferentes algoritmos. Realizamos dos experimentos considerando las condiciones de parada principales utilizadas en la literatura. Este trabajo es una extensión del previo donde se consideran diferente número de núcleos a la hora de ejecutar en paralelo los algoritmos (en el estudio anterior siempre se usaron 32 núcleos).

En el primer experimento, se usó como condición de parada un número fijo de evaluaciones de funciones, que es un esquema ampliamente usado. En la Figura A.9 representamos de forma visual los resultados del consumo energético de los algoritmos considerados.

La figura A.9 muestra que la utilización de más núcleos en las versiones distribuidas de los algoritmos redujo en gran medida el consumo de energía del proceso de búsqueda. Esta alta reducción del consumo de energía se produce debido a la cooperación de los algoritmos distribuidos en la ejecución del algoritmo, lo que permite alcanzar el límite de evaluaciones más rápido. Las versiones distribuidas implican la migración entre los algoritmos distribuidos, la migración reduce el esfuerzo de búsqueda al compartir el conocimiento a través de los diferentes núcleos de trabajo. Este intercambio de conocimiento de comunicación y búsqueda entre los algoritmos dará a los algoritmos distribuidos otro comportamiento de búsqueda diferente de los algoritmos secuenciales originales. Por lo tanto, el comportamiento de consumo de energía de los algoritmos distribuidos será diferente (generalmente mejor) de los secuenciales.
El segundo experimento difiere del anterior en la condición de parada y en este caso todos los algoritmos deben llegar al mismo valor específico de aptitud. Para una mejor comprensión de

UNIVERSIDAD DE MÁLAGA



(a) P-PEAKS20 (b) P-PEAKS50 (c) P-PEAKS100

(d) VRP1 (e) VRP2 (f) VRP3

(g) VRP4 (h) VRP5

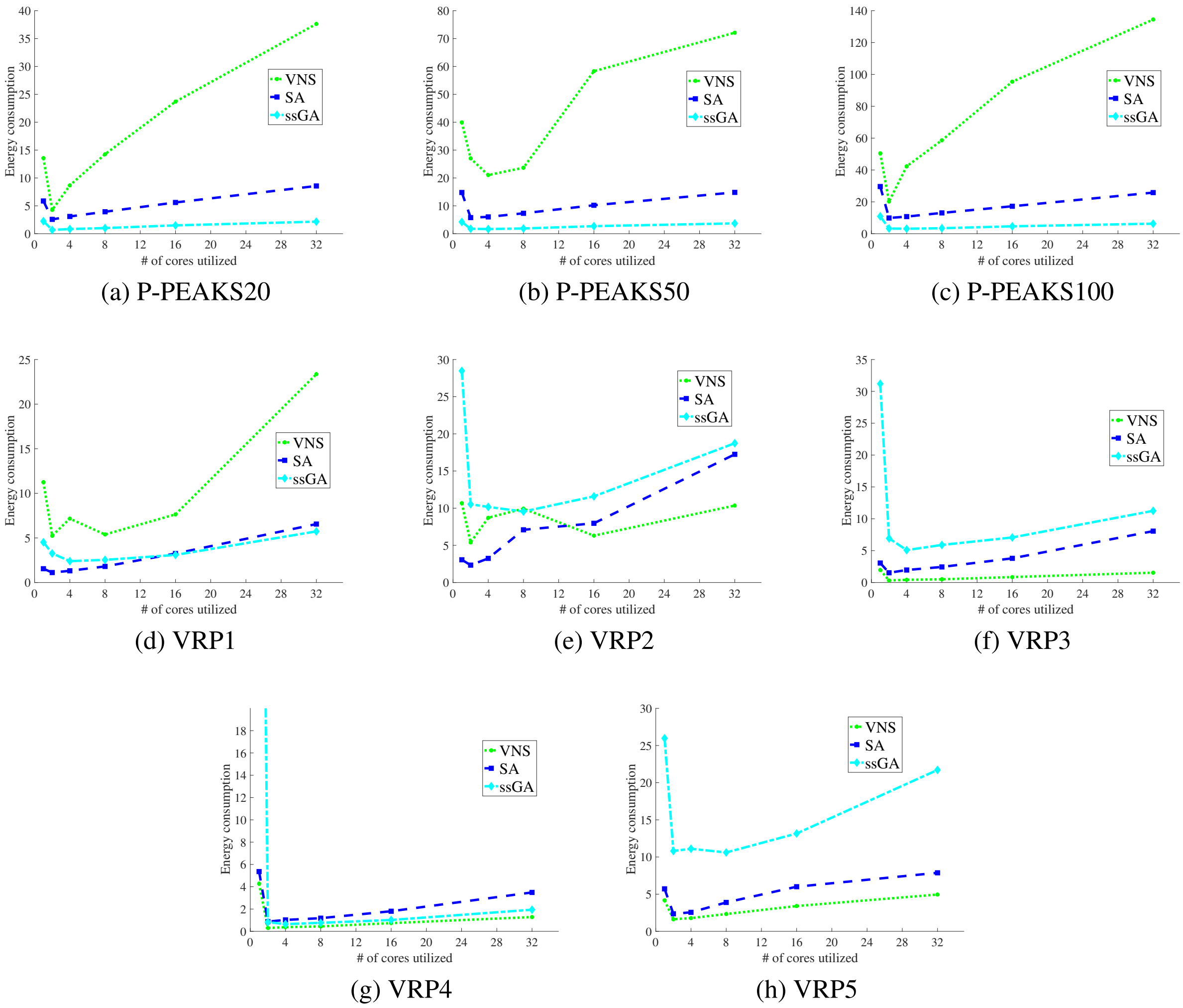
Figure A.9: Consumo energético respecto al número de núcleos usados (Exp. 1)

los resultados numéricos, presentamos una comparación visual de los diferentes algoritmos en la Figura A.10.

La utilización de más núcleos (dos y cuatro núcleos en nuestro caso) disminuyó el consumo de energía de todos nuestros algoritmos distribuidos en comparación con los secuenciales. A pesar de emplear una población de soluciones por cada paso algorítmico, las versiones distribuidas de ssGA mostraron un comportamiento competitivo de consumo energético en comparación con los algoritmos basados en trayectoria, que solo manejan una única solución. VNS y sus homólogos distribuidos consumieron los valores de energía más altos para las tres instancias P-PEAKS y VRP1. P-PEAKS es una función multimodal, por lo que el VNS básico (sin búsqueda local) requiere un mayor esfuerzo numérico para encontrar la solución óptima. Los algoritmos dSA con 2, 4, 8 y 16 núcleos obtuvieron mejores valores de consumo de energía en las tres instancias P-PEAKS y VRP4 en comparación con las secuenciales.

El resultado combinado de nuestros dos experimentos muestran el rendimiento numérico, el consumo de energía y el tiempo de ejecución de los algoritmos de optimización. Nuestros

UNIVERSIDAD DE MÁLAGA



(a) P-PEAKS20 (b) P-PEAKS50 (c) P-PEAKS100

(d) VRP1 (e) VRP2 (f) VRP3

(g) VRP4 (h) VRP5

Figure A.10: Consumo energético respecto al número de núcleos usados (Exp. 2)

comentarios sobre los resultados responden completamente a las preguntas de investigación propuestas en el artículo y confirman la relación entre la calidad de la solución, el consumo de energía y el tiempo de ejecución de los algoritmos considerados.

A.6 Conclusiones y Trabajo Futuro

Esta tesis presenta un amplio estudio sobre el rendimiento numérico y el consumo de energía de diferentes metaheurísticas. Medimos y analizamos el comportamiento del consumo energético y tiempo de ejecución de las metaheurísticas secuenciales y paralelas, dos paradigmas importantes de algoritmos de optimización, búsqueda y aprendizaje. Durante nuestros experimentos, hemos utilizado varios problemas (con diferentes características en su espacio de búsqueda y complejidad de la función de adecuación) en un número variado de núcleos y dimensiones para analizar el comportamiento de los algoritmos de forma exahustiva. Estas contribuciones satisfacen plenamente los objetivos planteados para el estudio del rendimiento de los algoritmos paralelos sobre

UNIVERSIDAD DE MÁLAGA

multiprocesadores. El resultado combinado de nuestros experimentos muestra el rendimiento numérico, el consumo de energía y el tiempo de ejecución tanto para versiones secuenciales como paralelas de los algoritmos de optimización. El resultado final de este estudio ayudará a mejorar el rendimiento de los algoritmos, reducir el consumo de energía y las emisiones de gases, lo cual es un desafío serio que requiere una mayor atención por parte de la sociedad.

También, investigamos el efecto del sincronismo en los GA paralelos sobre sistemas multiprocesadores, cubriendo los tres modelos paralelos más comunes: síncrono, asíncrono y maestro-esclavo. Nuestros experimentos muestran que la ganancia de velocidad podría ser alta e incluso superlineal en multiprocesadores para los diferentes algoritmos distribuidos probados. El modelo maestro-esclavo mostró un esfuerzo numérico competitivo con respecto al modelo basado en islas. El modelo maestro-esclavo demostró utilizar eficientemente los nuevos recursos de las unidades de procesamiento que están disponibles hoy en día, mientras que el diseño efectivo del modelo en islas permite explotar eficientemente el espacio de búsqueda. Además, realizamos un conjunto muy variado de experimentos para medir y analizar el comportamiento de consumo de energía y tiempo de GA y dGA utilizando diferentes problemas en un número creciente de núcleos y dimensiones para exponer el comportamiento potencial de los algoritmos. Observamos que el consumo de energía de los problemas varía de acuerdo con muchos factores, como el tamaño del problema, la complejidad de los operadores utilizados y los parámetros utilizados. Para los GAs secuenciales, los operadores genéticos y de aptitud consumen la mayor parte de la energía y el tiempo, mientras que el resto de las operaciones del algoritmo (mantenimiento) no requieren una cantidad significativa de energía en la mayoría de los escenarios.

Para analizar el rendimiento de las diferentes metaheurísticas, propusimos un análisis comparativo y combinatorio de ejecuciones distribuidas de algoritmos de optimización en un sistema multinúcleo. Para hacer una comparación justa, implementamos dos experimentos diferentes bajo los mismos requisitos (hardware, sistema operativo y lenguaje de programación). Los diversos experimentos realizados allí confirman la relación entre la calidad de la solución, el consumo de energía y el tiempo de ejecución de los algoritmos considerados. Bajo diferentes criterios de terminación, la eficiencia numérica de los algoritmos distribuidos fue razonable y competitiva en comparación con los algoritmos secuenciales.

Como líneas abiertas por esta tesis planteamos investigar la eficiencia de metaheurísticas paralelas sobre problemas de alta dimensión, considerando el ajuste de parámetros y empleando diferentes esquemas de comunicación. También se propone extender estos estudios de consumo de energía de tales técnicas para resolver problemas en el dominios de interés actual como las ciudades inteligentes. Planeamos proporcionar un marco general para diseñar metaheurísticas eficientes y conscientes de la energía. Los multiprocesadores se están volviendo muy útiles en este momento, y necesitamos más estudios sobre el comportamiento de las metaheurísticas en tales plataformas.

UNIVERSIDAD DE MÁLAGA

This page intentionally left blank.

UNIVERSIDAD DE MÁLAGA

Appendix B

List of Publications that Support the Work of the Thesis

In this Appendix, we present a list of publications that we have published during the study of this thesis. These publications are the result of the experiments that we have done to achieve the objectives of the thesis. These publications are published in JCR journals and international conferences. Detailed as follows:

B.1 Publications in JCR Journals:

[1] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors.** Swarm and Evolutionary Computation, 49:147-157, 2019.

Swarm and Evolutionary Computation:

- Impact factor: **6.33**
- Category: Computer Science, Artificial Intelligence, **[9/103] (Q1)**

[2] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. **A component-based study of energy consumption for sequential and parallel genetic algorithms.** The Journal of Supercomputing, 75(10):6194-6219, 2019.

The Journal of Supercomputing:

- Impact factor: **2.16**
- Category: Computer Science, Theory & Methods, **[44/103] (Q2)**

[3] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Parallel Execution Combinatorics with Metaheuristics: Comparative Study.** Swarm and Evolutionary Computation, 2020.
DOI: https://doi.org/10.1016/j.swevo.2020.100692

Swarm and Evolutionary Computation:

- Impact factor: **6.33**
- Category: Computer Science, Artificial Intelligence, **[9/103] (Q1)**

UNIVERSIDAD
DE MÁLAGA

B.2 Publications in Proceedings of International Conferences:

[4] Amr Abdelhafez and Enrique Alba. **Speed-up of synchronous and asynchronous distributed genetic algorithms: A first common approach on multiprocessors.** 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2677-2682, 2017.

- **Brief history of the congress**: The 19^{th} annual meeting of the IEEE Congress on Evolutionary Computation (CEC). CEC is one of the most important and leading conferences in evolutionary computation and computer science. The conference is technically sponsored by the Institute of Electrical and Electronics Engineers (IEEE), one of the largest technical association dedicated to developing technology in the world.
- Conference Rank: **B**, Source: CORE Rankings Portal - Computing Research & Education (CORE2017)
- Primary Field Of Research: Artificial Intelligence and Image Processing

[5] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. **Analyzing the energy consumption of sequential and parallel metaheuristics.** 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019.

- **Brief history of the congress**: The 17^{th} annual meeting of the International Conference on High-Performance Computing & Simulation (HPCS). The conference is targeting exploring original and state-of-the-art works in high performance and large scale computing systems. The conference is technically sponsored by the IEEE organization.
- Conference Rank: **B**, Source: CORE Rankings Portal - Computing Research & Education (CORE2018)
- Primary Field Of Research: Distributed Computing

UNIVERSIDAD DE MÁLAGA

References

[1] Amr Abdelhafez and Enrique Alba. Speed-up of synchronous and asynchronous distributed genetic algorithms: A first common approach on multiprocessors. *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2677–2682, 2017.

[2] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. A component-based study of energy consumption for sequential and parallel genetic algorithms. *The Journal of Supercomputing*, 75(10):1–26, 2019.

[3] Amr Abdelhafez, Enrique Alba, and Gabriel Luque. Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors. *Swarm and Evolutionary Computation*, 49:147–157, 2019.

[4] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. Analyzing the energy consumption of sequential and parallel metaheuristics. *2019 International Conference on High Performance Computing Simulation (HPCS)*, 2019.

[5] Amr Abdelhafez, Gabriel Luque, and Enrique Alba. Parallel execution combinatorics with metaheuristics: Comparative study. *Swarm and Evolutionary Computation*, 2020.

[6] S Burak Akat and Veysel Gazi. Decentralized asynchronous particle swarm optimization. In *2008 IEEE Swarm Intelligence Symposium*, pages 1–8. IEEE, 2008.

[7] Enrique Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, 82(1):7–13, 2002.

[8] Enrique Alba. *Parallel Metaheuristics: a new class of algorithms*. Wiley, 2005.

[9] Enrique Alba. *Optimization techniques for solving complex problems*. Wiley, 2009.

[10] Enrique Alba and Dorronsoro Bernabé. *Cellular genetic algorithms*. Springer, 2010.

[11] Enrique Alba, Francisco Luna, Antonio J Nebro, and José M Troya. Parallel heterogeneous genetic algorithms for continuous optimization. *Parallel Computing*, 30(5-6):699–719, 2004.

[12] Enrique Alba and José M. Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–52, 1999.

[13] Enrique Alba and José M. Troya. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Applied Intelligence*, 12(3):163–181, 2000.

[14] Enrique Alba and José M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17(4):451–465, 2001.

[15] Enrique Alba and José M. Troya. Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing*, 12(2):91–114, 2002.

[16] Josefa Díaz Álvarez, Juan Ángel García Martínez, Pedro Ángel Castillo Valdivieso, Fran-

UNIVERSIDAD
DE MÁLAGA

cisco Fernández de Vega, et al. Estimating energy consumption in evolutionary algorithms by means of frbs. In *EPIA Conference on Artificial Intelligence*, pages 229–240. Springer, 2017.

[17] Irma R Andalon-Garcia and Arturo Chavoya. Performance comparison of three topologies of the island model of a parallel genetic algorithm implementation on a cluster platform. In *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, pages 1–6. IEEE, 2012.

[18] Danial Jahed Armaghani, Mahdi Hasanipanah, Amir Mahdiyar, Muhd Zaimi Abd Majid, Hassan Bakhshandeh Amnieh, and Mahmood MD Tahir. Airblast prediction through a hybrid genetic algorithm-ann model. *Neural Computing and Applications*, 29(9):619–629, 2018.

[19] Dénes Bán, Rudolf Ferenc, István Siket, Ákos Kiss, and Tibor Gyimóthy. Prediction models for performance, power, and energy efficiency of software executed on heterogeneous hardware. *The Journal of Supercomputing*, 75(8):4001–4025, 2019.

[20] Rupayan Barua. Assessment and energy benchmarking for two archetype sustainable houses through comprehensive long term monitoring. *Assessment*, 1:1–2010, 2010.

[21] Plamenka Borovska. Efficiency of parallel metaheuristics for solving combinatorial problems. In *Proceedings of the 2007 international conference on Computer systems and technologies*, pages 1–6, 2007.

[22] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

[23] Jack Brimberg, Nenad Mladenović, Raca Todosijević, and Dragan Urošević. Solving the capacitated clustering problem with variable neighborhood search. *Annals of Operations Research*, 272(1-2):289–321, 2019.

[24] Guilherme Calandrini, Alfredo Gardel, Ignacio Bravo, Pedro Revenga, José L Lázaro, and F Javier Toledo-Moreo. Power measurement methods for energy efficient applications. *Sensors*, 13(6):7786–7796, 2013.

[25] Erick Cantu-Paz. *Efficient and accurate parallel genetic algorithms*, volume 1. Springer Science & Business Media, 2000.

[26] Erick Cantú-Paz. Master-slave parallel genetic algorithms. *Efficient and Accurate Parallel Genetic Algorithms Genetic Algorithms and Evolutionary Computation*, pages 33–48, 2001.

[27] Erick Cantu-Paz and David E Goldberg. Efficient parallel genetic algorithms: theory and practice. *Computer methods in applied mechanics and engineering*, 186(2-4):221–238, 2000.

[28] David G Carlson, Travis M Drucker, Timothy J Mullins, Jeffrey S McAllister, and Nelson Ramirez. Processing array data on simd multi-core processor architectures, July 9 2013. US Patent 8,484,276.

[29] Jing-fang Chen, Ling Wang, and Zhi-ping Peng. A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation*, 50:100557, 2019.

[30] Ling Chen, Hai-Ying Sun, and Shu Wang. A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem. *Infor-*

UNIVERSIDAD DE MÁLAGA

mation Sciences, 199:31–42, 2012.

[31] John Runwei Cheng and Mitsuo Gen. Accelerating genetic algorithms with gpu computing: A selective overview. *Computers & Industrial Engineering*, 128:514–525, 2019.

[32] Igor M Coelho, Vitor N Coelho, Eduardo J da S Luz, Luiz S Ochi, Frederico G Guimarães, and Eyder Rios. A gpu deep learning metaheuristic based model for time series forecasting. *Applied Energy*, 201:412–418, 2017.

[33] Teodor Gabriel Crainic and Michel Toulouse. Parallel strategies for meta-heuristics. *Handbook of Metaheuristics International Series in Operations Research & Management Science*, pages 475–513, 2003.

[34] José-Matías Cutillas-Lozano and Domingo Giménez. Optimizing shared-memory hyperheuristics on top of parameterized metaheuristics. In *ICCS*, volume 29, pages 20–29, 2014.

[35] Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. Rapl: memory power estimation and capping. In *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pages 189–194. IEEE, 2010.

[36] F Fernández de Vega, F Chávez, Javier Díaz, JA García, Pedro A Castillo, Juan J Merelo, and Carlos Cotta. A cross-platform assessment of energy consumption in evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 548–557. Springer, 2016.

[37] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N Suganthan, Carlos A Coello Coello, and Francisco Herrera. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, 48:220–250, 2019.

[38] Tansel Dokeroglu, Ender Sevinc, Tayfun Kucukyilmaz, and Ahmet Cosar. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137:106040, 2019.

[39] Bernabé Dorronsoro, Sergio Nesmachnow, Javid Taheri, Albert Y Zomaya, El-Ghazali Talbi, and Pascal Bouvry. A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing: Informatics and Systems*, 4(4):252–261, 2014.

[40] Marc Dubreuil, Christian Gagné, and Marc Parizeau. Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(1):229–235, 2006.

[41] Joan Escamilla, Miguel A Salido, Adriana Giret, and Federico Barber. A metaheuristic technique for energy-efficiency in job-shop scheduling. *The Knowledge Engineering Review*, 31(5):475–485, 2016.

[42] Ahmed Fanfakh, Jean-Claude Charr, Raphaël Couturier, and Arnaud Giersch. Energy consumption reduction for asynchronous message-passing applications. *The Journal of Supercomputing*, 73(6):2369–2401, 2016.

[43] Mitsuo Gen and Lin Lin. Genetic algorithms. *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

[44] Pawel Gepner and Michal Filip Kowalik. Multi-core processors: New way to achieve high system performance. In *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, pages 9–13. IEEE, 2006.

UNIVERSIDAD DE MÁLAGA

[45] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

[46] Dominik Goeke. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research*, 278(3):821–836, 2019.

[47] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, New York, 1989.

[48] Kayo Gonçalves-e Silva, Daniel Aloise, and Samuel Xavier-de Souza. Parallel synchronous and asynchronous coupled simulated annealing. *The Journal of Supercomputing*, 74(6):2841–2869, 2018.

[49] Guiliang Gong, Raymond Chiong, Qianwang Deng, Wenwu Han, Like Zhang, Wenhui Lin, and Kexin Li. Energy-efficient flexible flow shop scheduling with worker flexibility. *Expert Systems with Applications*, 141:112902, 2020.

[50] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300, 2015.

[51] Aljoscha Gruler, Javier Panadero, Jesica de Armas, José A Moreno Pérez, and Angel A Juan. A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research*, 2018.

[52] Rajat Gupta and Matt Gregg. Empirical evaluation of the energy and environmental performance of a sustainably-designed but under-utilised institutional building in the uk. *Energy and Buildings*, 128:68–80, 2016.

[53] Cristina Guzman, Alben Cardenas, and Kodjo Agbossou. Evaluation of meta-heuristic optimization methods for home energy management applications. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pages 1501–1506. IEEE, 2017.

[54] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A Moreno Pérez. Variable neighborhood search. In *Handbook of metaheuristics*, pages 57–97. Springer, 2019.

[55] Tomohiro Harada and Keiki Takadama. Performance comparison of parallel asynchronous multi-objective evolutionary algorithm with different asynchrony. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1215–1222. IEEE, 2017.

[56] Alberto Herrán, J Manuel Colmenar, and Abraham Duarte. A variable neighborhood search approach for the hamiltonian p-median problem. *Applied Soft Computing*, 80:603–616, 2019.

[57] Mark D Hill and Michael R Marty. Amdahl's law in the multicore era. *Computer*, 41(7):33–38, 2008.

[58] Abram Hindle. Green software engineering: The curse of methodology. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 5, pages 46–55, March 2016.

[59] Johannes Hofmann, Steffen Limmer, and Dietmar Fey. Performance investigations of genetic algorithms on graphics cards. *Swarm and Evolutionary Computation*, 12:33–47, 2013.

[60] Andy Hooper. Green computing. *Communication of the ACM*, 51(10):11–13, 2008.

[61] Toshihide Ibaraki, Koji Nonobe, and Mutsunori Yagiura. *Metaheuristics:: Progress as Real Problem Solvers*, volume 32. Springer Science & Business Media, 2006.

UNIVERSIDAD DE MÁLAGA

[62] Santiago Iturriaga, Sergio Nesmachnow, Bernabé Dorronsorro, and Pascal Bouvry. Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. *Computing and Informatics*, 32(2):273–294, 2013.

[63] Kenneth A Jong, Mitchell A Potter, and William M Spears. Using problem generators to explore the effects of epistasis. *The Seventh International Conference on Genetic Algorithms*, pages 338–345, 1997.

[64] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. Rapl in action. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 3(2):1–26, 2018.

[65] Kashif Nizam Khan, Zhonghong Ou, Mikael Hirki, Jukka K Nurminen, and Tapio Niemi. How much power does your server consume? estimating wall socket power using rapl measurements. *Computer Science-Research and Development*, 31(4):207–214, 2016.

[66] Dong-Sun Kim, Hyun-Sik Kim, Youn-Sung Lee, and Duck-Jin Chung. On the design of a parallel genetic algorithm based on a modified survival method for evolvable hardware. *Computational Intelligence and Bioinspired Systems Lecture Notes in Computer Science*, pages 541–551, 2005.

[67] Hideaki Kimura, Mitsuhisa Sato, Yoshihiko Hotta, Taisuke Boku, and Daisuke Takahashi. Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster. In *2006 IEEE international conference on cluster computing*, pages 1–10. IEEE, 2006.

[68] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220:621–630, 1983.

[69] Slawomir Koziel and Xin-She Yang. *Computational Optimization, Methods and Algorithms*. Springer Berlin Heidelberg, 2011.

[70] Soniya Lalwani, Harish Sharma, Suresh Chandra Satapathy, Kusum Deep, and Jagdish Chand Bansal. A survey on parallel particle swarm optimization algorithms. *Arabian Journal for Science and Engineering*, 44(4):2899–2923, 2019.

[71] Hooi Hooi Lean and Russell Smyth. Co2 emissions, electricity consumption and output in asean. *Applied Energy*, 87(6):1858–1864, 2010.

[72] Sangmin Lee and Seoung Bum Kim. Parallel simulated annealing with a greedy algorithm for bayesian network structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[73] Ming Li, Deming Lei, and Jingcao Cai. Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. *Swarm and Evolutionary Computation*, 49:34–43, 2019.

[74] Yan Y. Liu and Shaowen Wang. A scalable parallel genetic algorithm for the generalized assignment problem. *Parallel Computing*, 46:98–119, 2015.

[75] Yan Y Liu and Shaowen Wang. A scalable parallel genetic algorithm for the generalized assignment problem. *Parallel computing*, 46:98–119, 2015.

[76] Jia Luo, Shigeru Fujimura, Didier El Baz, and Bastien Plazolles. Gpu based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem. *Journal of Parallel and Distributed Computing*, 133:244–257, 2019.

[77] Haiping Ma, Shigen Shen, Mei Yu, Zhile Yang, Minrui Fei, and Huiyu Zhou. Multi-

UNIVERSIDAD DE MÁLAGA

population techniques in nature inspired optimization algorithms: a comprehensive survey. *Swarm and evolutionary computation*, 44:365–387, 2019.

[78] Gonzalo Martín, David E Singh, Maria-Cristina Marinescu, and Jesús Carretero. Enhancing the performance of malleable mpi applications by using performance-aware dynamic reconfiguration. *Parallel Computing*, 46:60–77, 2015.

[79] Matthew T. Mcmahon and Layne T. Watson. A distributed genetic algorithm with migration for the design of composite laminate structures. *Parallel Algorithms and Applications*, 14(4):329–362, 2000.

[80] Suejb Memeti, Sabri Pllana, Alécio Binotto, Joanna Kołodziej, and Ivona Brandic. Using meta-heuristics and machine learning for software optimization of parallel computing systems: a systematic literature review. *Computing*, 101(8):893–936, 2019.

[81] Mohand Mezmaz, Nouredine Melab, Yacine Kessaci, Young Choon Lee, E-G Talbi, Albert Y Zomaya, and Daniel Tuyttens. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *Journal of Parallel and Distributed Computing*, 71(11):1497–1508, 2011.

[82] Efstathios E. (Stathis) Michaelides. Environmental and ecological effects of energy production and consumption. *Green Energy and Technology Alternative Energy Sources*, pages 33–63, 2012.

[83] Nenad Mladenović, Milan Dražić, Vera Kovačevic-Vujčić, and Mirjana Čangalović. General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3), 2008.

[84] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[85] Sanaz Mostaghim, Jurgen Branke, Andrew Lewis, and Hartmut Schmeck. Parallel multi-objective optimization using master-slave model on heterogeneous resources. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1981–1987. IEEE, 2008.

[86] Lluís-Miquel Munguía, Shabbir Ahmed, David A. Bader, George L. Nemhauser, and Yufen Shao. Alternating criteria search: a parallel large neighborhood search algorithm for mixed integer programs. *Computational Optimization and Applications*, 69(1):1–24, Aug 2018.

[87] Payam Nejat, Fatemeh Jomehzadeh, Mohammad Mahdi Taheri, Mohammad Gohari, and Muhd Zaimi Abd. Majid. A global review of energy consumption, co_2 emissions and policy in the residential sector (with an overview of the top ten co_2 emitting countries). *Renewable and Sustainable Energy Reviews*, 43:843–862, 2015.

[88] Krisanarach Nitisiri, Mitsuo Gen, and Hayato Ohwada. A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling. *Computers & Industrial Engineering*, 130:381–394, 2019.

[89] Panos M Pardalos and H Edwin Romeijn. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013.

[90] Martín Pedemonte, Francisco Luna, and Enrique Alba. A theoretical and empirical study of the trajectories of solutions on the grid of systolic genetic search. *Information Sciences*, 445:97–117, 2018.

[91] Jun Pei, Nenad Mladenović, Dragan Urošević, Jack Brimberg, and Xinbao Liu. Solving the

UNIVERSIDAD DE MÁLAGA

traveling repairman problem with profits: A novel variable neighborhood search approach. *Information Sciences*, 507:108–123, 2020.

[92] Jinqing Peng, Lin Lu, Hongxing Yang, and Jun Han. Investigation on the annual thermal performance of a photovoltaic wall mounted on a multi-layer façade. *Applied energy*, 112:646–656, 2013.

[93] Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. Energy efficiency across programming languages: how do energy, time, and memory relate? In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, pages 256–267, 2017.

[94] Carsten Peterson. Parallel distributed approaches to combinatorial optimization: benchmark studies on traveling salesman problem. *Neural computation*, 2(3):261–269, 1990.

[95] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on mobile computing*, 5(2):128–143, 2005.

[96] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Parallel pareto optimization for subset selection. In *IJCAI*, pages 1939–1945, 2016.

[97] Yuzhuo Qiu, Liang Wang, Xiaoling Xu, Xuanjing Fang, and Panos M Pardalos. A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, 66:311–318, 2018.

[98] Juan Rada-Vilela, Mengjie Zhang, and Winston Seah. A performance study on synchronicity and neighborhood size in particle swarm optimization. *Soft Computing*, 17(6):1019–1030, 2013.

[99] Thomas Rauber, Gudula Rünger, Michael Schwind, Haibin Xu, and Simon Melzner. Energy measurement, modeling, and prediction for processors with frequency scaling. *The Journal of Supercomputing*, 70(3):1451–1476, 2014.

[100] James Reinders. *Intel threading building blocks: outfitting C++ for multi-core processor parallelism.* O'Reilly Media, Inc., 2007.

[101] Ana Carolina Riekstin, Bruno Bastos Rodrigues, Kim Khoa Nguyen, Tereza Cristina Melo De Brito Carvalho, Catalin Meirosu, Burkhard Stiller, and Mohamed Cheriet. A survey on metrics and measurement tools for sustainable distributed cloud networks. *IEEE Communications Surveys & Tutorials*, 20(2):1244–1270, 2018.

[102] Eyder Rios, Luiz Satoru Ochi, Cristina Boeres, Vitor N Coelho, Igor M Coelho, and Ricardo Farias. Exploring parallel multi-gpu local search strategies in a metaheuristic framework. *Journal of Parallel and Distributed Computing*, 111:39–55, 2018.

[103] Wilson Rivera. Scalable parallel genetic algorithms. *Artificial Intelligence Review*, 16(2):153–168, 2001.

[104] Vincent Roberge, Mohammed Tarbouchi, and Francis Okou. Optimal power flow based on parallel metaheuristics for graphics processing units. *Electric Power Systems Research*, 140:344–353, 2016.

[105] Manuel Rodríguez-Gonzalo, David E Singh, Javier García Blas, and Jesús Carretero. Improving the energy efficiency of mpi applications by means of malleability. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 627–634. IEEE, 2016.

UNIVERSIDAD DE MÁLAGA

[106] Catherine Roucairol. Parallel processing for difficult combinatorial optimization problems. *European Journal of Operational Research*, 92(3):573–590, 1996.

[107] M. Ruciński, D. Izzo, and F. Biscani. On the impact of the migration topology on the island model. *Parallel Computing*, 36(10-11):555–571, 2010.

[108] Matt Ryerkerk, Ron Averill, Kalyanmoy Deb, and Erik Goodman. A survey of evolutionary algorithms using metameric representations. *Genetic Programming and Evolvable Machines*, 2019.

[109] Sergio Santander-Jiménez and Miguel A Vega-Rodríguez. Asynchronous non-generational model to parallelize metaheuristics: a bioinformatics case study. *IEEE Transactions on Parallel and Distributed Systems*, 28(7):1825–1838, 2017.

[110] Andrea Serani, Cecilia Leotardi, Umberto Iemma, Emilio F Campana, Giovanni Fasano, and Matteo Diez. Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydrodynamics problems. *Applied Soft Computing*, 49:313–334, 2016.

[111] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[112] S. Tamilselvi, S. Baskar, L. Anandapadmanaban, V. Karthikeyan, and S. Rajasekar. Multi objective evolutionary algorithm for designing energy efficient distribution transformers. *Swarm and Evolutionary Computation*, 42:109–124, 2018.

[113] Alexandru-Adrian Tantar, Nouredine Melab, and El-Ghazali Talbi. A comparative study of parallel metaheuristics for protein structure prediction on the computational grid. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–10. IEEE, 2007.

[114] Diego Teijeiro, Xoán C Pardo, Patricia González, Julio R Banga, and Ramón Doallo. Towards cloud-based parallel metaheuristics: a case study in computational biology with differential evolution and spark. *The International Journal of High Performance Computing Applications*, 32(5):693–705, 2018.

[115] Andrea Toselli and Olof Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2006.

[116] Anne E. Trefethen and Jeyarajan Thiyagalingam. Energy-aware software: Challenges, opportunities and strategies. *Journal of Computational Science*, 4(6):444–449, 2013.

[117] Chun-Wei Tsai, Shi-Jui Liu, and Yi-Chung Wang. A parallel metaheuristic data clustering framework for cloud. *Journal of Parallel and Distributed Computing*, 116:39–49, 2018.

[118] Akshay Venkatesh, Krishna Kandalla, and Dhabaleswar K Panda. Evaluation of energy characteristics of mpi communication primitives with rapl. In *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, pages 938–945. IEEE, 2013.

[119] Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. Parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. *Journal of Aerospace Computing, Information, and Communication*, 3(3):123–137, 2006.

[120] Xiaoyong Xu, Maolin Tang, and Yu-Chu Tian. Theoretical results of qos-guaranteed resource scaling for cloud-based mapreduce. *IEEE Transactions on Cloud Computing*, 6(3):879–889, 2016.

UNIVERSIDAD
DE MÁLAGA

[121] Xingjian Xu, Zhaohua Ji, Fangfang Yuan, and Xiaoqin Liu. A novel parallel approach of cuckoo search using mapreduce. In *2014 International Conference on Computer, Communications and Information Technology (CCIT 2014)*. Atlantis Press, 2014.

[122] Zefeng Xu and Yanguang Cai. Variable neighborhood search for consistent vehicle routing problem. *Expert Systems with Applications*, 113:66–76, 2018.

[123] Alexandru-Ciprian Zavoianu, Susanne Saminger-Platz, and Wolfgang Amrhein. Comparative analysis of two asynchronous parallelization variants for a multi-objective coevolutionary solver. *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019.

[124] Zhi-Hui Zhan, Xiao-Fang Liu, Huaxiang Zhang, Zhengtao Yu, Jian Weng, Yun Li, Tianlong Gu, and Jun Zhang. Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version. *IEEE Transactions on Parallel and Distributed Systems*, 28(3):704–716, 2016.

[125] Junhao Zhou, Hong Xiao, Hao Wang, and Hong-Ning Dai. Parallelizing simulated annealing algorithm in many integrated core architecture. *Computational Science and Its Applications - ICCSA 2016 Lecture Notes in Computer Science*, pages 239–250, 2016.

UNIVERSIDAD
DE MÁLAGA